IDA PAPER P-2172

REVIEWS OF SELECTED SYSTEM AND SOFTWARE TOOLS FOR STRATEGIC DEFENSE APPLICATIONS

David A. Wheeler
Dennis W. Fife
Edgar H. Sibley
J. Bret Michael

February 1990

Prepared for
Strategic Defense Initiative Organization

INSTITUTE FOR DEFENSE ANALYSES
1801 N. Beauregard Street, Alexandria, Virginia 22311-1772

IDA Log No. HQ 89-053676

DEFINITIONS

IDA publishes the following documents to report the results of its work.

Reports

Reports are the most authoritative and most carefully considered products IDA publishes. They normally embody results of major projects which (a) have a direct bearing on decisions affecting major programs, (b) address issues of significant concern to the Executive Branch, the Congress and/or the public, or (c) address issues that have significant economic implications. IDA Reports are reviewed by outside panels of experts to ensure their high quality and relevance to the problems studied, and they are released by the President of IDA.

Group Reports

Group Reports record the findings and results of IDA established working groups and panels composed of senior individuals addressing major issues which otherwise would be the subject of an IDA Report. IDA Group Reports are reviewed by the senior individuals responsible for the project and others as selected by IDA to ensure their high quality and relevance to the problems studied, and are released by the President of IDA.

Papers

Papers, also authoritative and carefully considered products of IDA, address studies that are narrower in scope than those covered in Reports. IDA Papers are reviewed to ensure that they meet the high standards expected of refereed papers in professional journals or formal Agency reports.

Documents

IDA Documents are used for the convenience of the sponsors or the analysts (a) to record substantive work done in quick reaction studies, (b) to record the proceedings of conferences and meetings, (c) to make available preliminary and tentative results of analyses, (d) to record data developed in the course of an investigation, or (e) to forward information that is essentially unanalyzed and unevaluated. The review of IDA Documents is suited to their content and intended use.

The work reported in this document was conducted under contract MDA 903 84 C 0031 for the Department of Defense. The publication of this IDA document does not indicate endorsement by the Department of Defense, nor should the contents be construed as reflecting the official position of that Agency.

This Paper has been reviewed by IDA to assure that it meets high standards of thoroughness, objectivity, and appropriate analytical methodology and that the results, conclusions and recommendations are properly supported by the material presented.

Approved for public release, unlimited distribution. Unclassified.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	February 1990	Final

4. TITLE AND SUBTITLE
Reviews of Selected System and Software Tools for Strategic Defense Applications

5. FUNDING NUMBERS
MDA 903 89 C 0003

T-R2-597.2

6. AUTHOR(S)
Dennis W. Fife, Edgar H. Sibley, J. Bret Michael, David A. Wheeler

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Institute for Defense Analyses (IDA)
1801 N. Beauregard Street
Alexandria, VA 22311-1772

8. PERFORMING ORGANIZATION REPORT NUMBER
IDA Paper P-2177

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Strategic Defense Initiative Organization (SDIO)
SDIO/ENA
Room 3E149, The Pentagon
Washington, D.C. 20301-7100

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Public release/unlimited distribution.

12b. DISTRIBUTION CODE
2A

13. ABSTRACT (Maximum 200 words)

Information about certain software engineering tools relevant to the needs of the Strategic Defense Initiative (SDI) program is described in this paper. A broad sample of available off-the-shelf tools and their basic function and scope are provided. The products reviewed are used for requirements analysis and preliminary design. These tools fall into the category of computer-aided software (or systems) engineering (CASE).

14. SUBJECT TERMS
Software Engineering Tools; Strategic Defense Initiative (SDI); Computer-Aided Software Engineering (CASE); Requirements Analysis; Preliminary Design.

15. NUMBER OF PAGES
90

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	SAR

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

IDA PAPER P-2177

REVIEWS OF SELECTED SYSTEM AND SOFTWARE TOOLS FOR STRATEGIC DEFENSE APPLICATIONS

David A. Wheeler
Dennis W. Fife
Edgar H. Sibley
J. Bret Michael

February 1990

INSTITUTE FOR DEFENSE ANALYSES
Contract MDA 903 89 C 0003
Task T-R2-597.2

TABLE OF CONTENTS

TRADEMARK ACKNOWLEDGEMENTS

All terms mentioned in this paper that are known to be trademarks or service marks are listed below. Use of a term in this paper should not be regarded as affecting the validity of any trademark or service mark.

Apple is a registered trademark of Apple Computer, Inc.
Context is a registered trademark of Mentor Graphics.
DEC, VMS, ULTRIX, and VAXstation are registered trademarks of Digital Equipment Corporation.
FMap and TMap are trademarks of Hamilton Technologies, Inc.
IBM is a registered trademark of International Business Machines Corp.
Macintosh is a trademark licensed to Apple Computer, Inc.
Microsoft and MS-DOS are registered trademarks of Microsoft Corp.
PostScript is a trademark of Adobe Systems, Incorporated.
UNIX is a registered trademark of AT&T.

PREFACE

IDA Paper P-2177, *Reviews of Selected System and Software Engineering Tools for Strategic Defense Applications*, was prepared for the Strategic Defense Initiative Organization (SDIO) in response to tasking contained in IDA Task Order T-R2-597.2 under contract MDA 903-89-C-0003.

This paper is intended to be a supplement to IDA Paper P-2062, *Evaluation of Computer-Aided System Design Tools for SDI Battle Management/C3 Architecture Development*, dated October 1987. The current document provides updated information on the tools evaluated in the previous document plus a number of additional tools not covered in the previous document.

The authors wish to thank the many reviewers within IDA: Terry Mayfield, James Baldo, Cathy Jo Linn, Howard Cohen, and Debbie Heystek. We also express our appreviation to Sylvia Reynolds for her editorial advice and assistance, and to Donna Graham for her support in formatting the document.

1. INTRODUCTION

1.1 PURPOSE

This paper provides current information about certain software engineering tools relevant to the needs of the Strategic Defense Initiative (SDI) program. It partially fulfills requirements of IDA Task T-R2-597.2. The paper covers a broad sample of available, off-the-shelf tools, and describes their basic functions and scope. The information is for technical introduction and current awareness of this field, not for assessing effectiveness of any tool or for competitively selecting one tool from among those available. In choosing the tools to be covered, IDA has considered SDI needs for rigorous specification, thorough diagnostics to validate design, and the possibility of automatic code generation for prototype validation.

1.2 SCOPE

The paper provides information on selected commercial tools which, for various reasons, could not be covered in an earlier evaluation [Fife 87]. It also discusses recent enhancements and other information on the tools that were previously evaluated.

The products that were reviewed are used for requirements analysis and preliminary design. These tools fall into the category that is often called computer-aided software (or systems) engineering (CASE).

1.3 APPROACH

The following tools are reviewed in the order shown and except for DCDS/Ada, all product names are trademarks of the companies indicated:

Product	**Vendor/Sponsor**
Software Through Pictures	Interactive Development Environments, Inc.
Teamwork	Cadre Technologies, Inc.
TAGS	Teledyne Brown Engineering

Auto-G	Advanced System Architectures
DCDS/Ada	TRW/U. S. Army Strategic Defense Command
RDD-100	Ascent Logic Corp.
STATEMATE	i-Logix, Inc.
Refine	Reasoning Systems, Inc.
Spectrum	Software Architecture and Engineering, Inc.
Design/family (incl., Design/IDEF)	Meta Software Corp.
001	Hamilton Technologies, Inc.
Foresight	Athena Systems, Inc.
VSF	Systematica Limited
Adagen	Mark V Systems Limited

Inclusion of a tool in this report does not imply a recommendation for its use in any particular role for SDI. The reviews do not include the many PC-based CASE tools already established in commercial software practice, most of which are oriented toward dataflow diagramming. Such tools are not considered to have high potential for SDI applications because of their limitations, especially in graphics and specification depth.

Each tool was reviewed relative to a number of requirements areas established in the previous report. The identifying heading and a brief scope statement for each are listed below.

Heading	**Scope**
Graphics and Editing	User interface characteristics, including ease of use
Design Semantics and Support	Specification depth for fully specifying real-time defense systems
Team Design Support	Supporting team projects, including configuration management, etc.
Documentation and Output	Reporting and documenting design results
Static Diagnostics	Testing design completeness and correctness, excluding behavior
Simulation	Simulating the designed system's behavior to validate design
Adaptability	Extending or tailoring the tool for special needs
Interoperability	Using the tool or its output with other tools
Traceability	Relating design results to original requirements
Information Modeling Support	Representing both the designed system's data and design semantics
Distinguishing Capabilities	Summary of major distinctive features

Each review is based upon IDA analysis, tool manuals and technical papers, discussions with vendors, and tool demonstrations. The significance and complexity of some tools warrants more analysis than could be done for this report.

1.4 CONCLUSIONS

This paper does not seek to define which tools are suitable for specific tasks, or which meet SDI requirements, or in general, which tool is best for a certain task.

The following offers one way to categorize the tools according to their general usage and state of development:

a. Well known tools, previously reviewed, with strong commercial and government clientele and significant usage on large defense applications: TAGS, DCDS/Ada, Auto-G, Software Through Pictures, and Teamwork;

b. Tools more recently released that have gained significant interest and use for defense applications and are becoming strong competitors of the group above: RDD, Foresight, Design/family, and STATEMATE;

c. Promising tools which are available, though still undergoing research and development to complete functionality that would be important to SDI applications: Adagen, 001, Refine, Spectrum, and VSF.

IDA has previously reported a wide range of differences in technique, functionality, and potential effectiveness among CASE tools [Fife 87]. The tool reviews in this report, though not as detailed as the earlier report, illustrate that more capable tools are emerging for SDI consideration than were available previously. The well known tools are being enhanced, and prospects for other innovative or unique tools are very encouraging.

Some tools appear to have advantages for particular kinds of technical problems or project situations. An advantage, for example, may be familiarity to certain technical personnel, a certain kind of documentation output, ease of validating a certain engineering approach, or the suitability of a tool's design method for a key problem such as security. Provided that the costs of training and initial data capture are acceptable, SDI should exploit the unique advantages that different tools may offer in filling niches within the scope of SDI needs.

A number of other conclusions can be drawn from the capabilities and enhancements seen in the group of tools included in this review:

a. Deficiencies of typical dataflow diagramming tools for real-time defense applications, which IDA previously noted, appear to be recognized and are leading to enhancements that generalize specification capabilities.

b. Simulation capability clearly is gaining importance as a means to extend diagnostic and validation support given by tools. Auto-G produces SADMT [Linn

88], TAGS and DCDS/Ada have their own simulation components, and STATEMATE, RDD, Teamwork, Foresight, Design/family, Refine, and Spectrum have, or soon will have, either simulation or animation features.

c. Capability to automatically generate code suitable for prototype evaluation is demonstrable, though overall quality and specific limitations need further evaluation. Auto-G, TAGS, STATEMATE, and Adagen have demonstrated Ada code generation. Auto-G also generates C code. Refine and Spectrum, generating LISP and C code respectively, also illustrate commitment to code generation from a tool. The goal is to have a tool produce complete, compilable programs, as opposed to producing just package specifications or program skeletons which programmers then must complete manually. A designer must specify software components and behavior fully at an algorithmic level in order to fully generate code. A tool must have the specification depth to support this design effort.

d. Automated documentation support and an interface to electronic publishing software, covering MIL-STD-2167A or user-defined standards, has become a typical tool capability. TAGS, RDD, DCDS/Ada, Teamwork, Software Through Pictures, and STATEMATE provide documentation templates.

e. Availability of a tool on several popular workstations is becoming commonplace. Teamwork and Software Through Pictures previously demonstrated this, and now TAGS, RDD, Design/family, STATEMATE, and Adagen are available on several workstations. DCDS/Ada is now available on Sun workstations.

f. Configuration management capabilities are gaining attention for assisting teams on large projects. TAGS has been notable for its features in this area. Cadre recently added Model Configuration Management to Teamwork, and DCDS also has added configuration management features. RDD provides features to assist merging and splitting of databases.

g. User interface and graphic sophistication has improved. STATEMATE exhibits an effective color interface. RDD demonstrates a specialized user interface. In its Sun implementation DCDS/Ada has improved its user interface.

h. Real-time system specification requires that both the logical and temporal aspects of a system be taken into account. There is no wide agreement on the effectiveness of various specification techniques for real-time requirements.

i. Most of the tools offer minimal compatibility among the design data produced by each of their tool components. Despite their dependence on database capability, only a few tools, such as TAGS and Software Through Pictures, have a high-level query language applicable to all data. DCDS/Ada has a single high level query language, but the language can only be applied to one of five specification stages at a time. RDD has a report generation language but not a query language. For many tools, design database semantics are neither extendible nor visible. This limits a designer's ability to fully exploit the entire database to understand and validate all aspects of a designed system.

j. Many of the tools do not have an open method for bidirectional transfer of data between themselves and other tools. Software through Pictures, DCDS, and TAGS do have such methods for importing and exporting design data.

k. Any standard toolset chosen for SDI should provide flexibility to exploit the strengths or innovations of different tools. This assigns a high level of importance to capabilities for transferring design data between tools.

1.5 SUMMARY REVIEWS

The remainder of the report comprises the tool summaries. The tools reviewed in [Fife 87] are listed first, followed by the additional tools reviewed in this paper.

APPENDIX A

SOFTWARE THROUGH PICTURES

Software through Pictures (StP) is a dataflow-oriented software engineering tool, developed by Interactive Development Environments, San Francisco, CA [IDE 88]. This review concerns the features of the most current version, Release 4.0. The major recent additions to StP are: an object annotation editor, DOD-STD-2167A document templates and a template editor for automating standardized reports, and an interface to the Interleaf Technical Publishing Software.

Graphics and Editing

StP provides graphic and text editors that take advantage of the windowing capabilities of the host system and that employ static and pull-down menus, keyboard, and mouse.

Design Semantics and Support

StP supports the dataflow diagramming method with control flow and state transition diagrams or tables for extending design to the specification of real-time systems. Entity relationship and software module structure diagrams are included.

StP's new Object Annotation Editor (OAE) serves to associate properties and values with any object in a diagram or with an entire diagram. The annotation information is stored in the data dictionary. The OAE is invoked by a menu choice, followed by a mouse pick on the desired object. OAE guides the user through the annotation entry procedure in a separate display window.

The approach is based on annotation templates provided for all types of diagram objects. A template is a blank annotation record, set up with predefined properties and a textual description field. Templates can be created or the predefined ones modified by a user via a text editor. This approach supports the capture of user-specified design information related to graphic objects.

Annotations are organized into "note" types. An object may have many note types. Each note type represents a set of related properties of an object. Each property is referred to as a note item, and a free-form text item is available.

Annotation information in the data dictionary may be extracted by the user with the Object Management Language, Document Preparation System, or the data

manipulation language of the Troll DBMS.

StP does not explicitly represent design object replication, time or other quantitative performance factors, or software/hardware resource allocation. Such items could be captured informally using the object annotation editor.

Team Design Support

StP supports multiuser development of software systems, relying on the host operating system's configuration management and file management facilities. For example, if StP is running under Unix, StP uses Unix's Source Code Control System for configuration and version management.

Documentation and Output

The contents of the text and graphics files created with StP can be printed in PostScript, Unix pic, or raster formats. The entire contents of the data dictionary may also be printed.

StP's Document Preparation System (DPS) provides a set of DOD-STD-2167 report templates. Users can create and modify their own custom report templates via the Documentation Definition Editor. Information in the data dictionary is inserted into the templates for generating reports. A document browsing capability is provided for viewing documents and for interfacing with external word processing systems. Currently, the only desktop publishing system that StP specifically supports is the Interleaf Technical Publishing Software. The DPS provides for mixing text and graphics.

StP will generate data declarations from the design description, but does not generate code. The user can key in process code as part of process specifications (P-Specs). StP treats these as textual descriptions.

Static Diagnostics

Static diagnostics available in StP include diagram and decomposition checking. These detect simple errors such as missing labels, missing decomposition connections, unconnected objects, etc. No enhancements are evident beyond the prior IDA report.

Simulation

No capability for design simulation or dynamic checking is provided.

Adaptability

No adaptability enhancements have been made regarding graphics, but the object annotation method is an important adaptability feature.

Interoperability

The Interleaf Technical Publishing System is the only package to which IDE provides explicit hooks. StP does provide data import and export facilities as part of the DBMS. The import and export data formats are well documented. Diagrams are stored as text, and the storage format for diagrams also is well documented.

Traceability

There is no automatic traceability capability. Object annotations can be used for tracing specifications and design objects back to requirements.

Information Modeling Support

StP provides ERA or Jackson data structure diagramming for graphically defining system data items, structures, and relationships. But, StP does not provide the key features needed for information modeling or database design. For example, StP does not support inheritance or computer-aided database normalization. StP itself is based on Troll, a relational database system which can be accessed separately.

Distinguishing Capabilities

The new object annotation feature is one of StP's most distinctive features. In addition, StP has a well designed and easy to use user interface. The tool forces the user to take a dataflow-oriented approach to building software systems. Likewise, the tool requires users to build their own configuration and project management facilities, hooks to other software products, and so on.

APPENDIX B

TEAMWORK

Teamwork is a product of Cadre Technologies, Inc., Providence, Rhode Island, and is available for Apollo, Sun, DEC VMS and ULTRIX workstations, Hewlett-Packard workstations, IBM OS/2 and AIX, IBM PC/RT, and other machines listed in the previous IDA report. Teamwork embodies the dataflow diagramming approach to system design, following the conventions of Yourdon and DeMarco. It provides the Hatley conventions depicting real-time considerations. Teamwork's components are:

a. Teamwork/SA, for structured analysis;

b. Teamwork/RT, extending SA for real-time considerations;

c. Teamwork/SD, for structured design;

d. Teamwork/IM, for database information modeling;

e. Teamwork/ACCESS, the database access utility;

f. Teamwork/DPI, the document production interface;

g. Teamwork/Menus, for tailoring or extending Teamwork menus;

h. Teamwork/Ada, a graphic editor for Ada program design;

i. Teamwork/ASB, an Ada source builder;

j. Teamwork/CSB, a C source builder.

Graphics and Editing

As IDA previously reported, Teamwork has an effective and easily used interface, based on fixed and pull-down menus, keyboard, and mouse selection and pointing. The graphics implementation was improved in performance during the last year, compared to the version IDA used in its hands-on evaluation.

Design Semantics and Support

A significant addition to Teamwork's components is Teamwork/Ada, a graphic editor for designing Ada program structures with the Buhr icons and object-oriented design approach. This tool is the first stage of a long range collaboration with General Electric Research and Development Center, by which the tools that GE developed in the

Ada Programmer's Workbench will be reimplemented as part of Teamwork.

Team Design Support

Teamwork's support for project teams and other distributed efforts has been enhanced with its Model Configuration Management (MCM) facility. MCM automatically generates and tags versions of diagrams, each with its own status label to capture the change author and history. A single command freezes and protects a project model in its entirety as a baseline. A project manager, as "owner" can control access privileges for read, write, or delete access, for owner, a group, or all Teamwork users. MCM also aids the merging of diagrams created by different individuals into one project model. Name conflicts are automatically detected during merging, and can be resolved interactively by one user.

Documentation and Output

A Graphic Note feature has been added to Teamwork's Annotate facility. The latter permits free-text notes to be attached to diagrams or objects. Now, a free-form line diagram also may form a note. These graphic diagrams also can be created as templates to be copied, pasted, and tailored as elements of larger diagrams. One or more graphic notes may be attached to elements within any structured analysis and design diagram, or to an entire diagram. The collection of existing notes for a project or object is viewed through an index. Individual notes may be chosen from the index for presentation on the user's display screen.

An SQL report writer may be used to extract data dictionary information.

The Teamwork Document Production Interface (DPI) provides templates for combining diagrams and text files to serve as first-cut versions of DoD MIL-STD-2167A documents. The output subsequently is manipulated through an external publishing software package, such as Interleaf, Scribe, or Context.

Static Diagnostics

Teamwork's diagnostics are typical of dataflow diagramming tools, but with added flexibility in applying them. They may be run in background mode, as well as the foreground, online mode. Also, a user may select to apply diagnostics to a single diagram, or to the set of diagrams decomposed from a given diagram, as well as to an entire project collection.

Simulation

A tool is available from Cadre which translates Teamwork data into ADAS, which can then perform simulations.

Adaptability

Cadre's approach to Teamwork adaptability rests on user-written programs to access Teamwork's database through the provided ACCESS package. This supports a variety of user-written analysis and reporting programs, and Cadre provides documented examples for a few major ones of interest. No changes to graphics are available, however.

Interoperability

As noted above, Teamwork provides its DPI output in a format for publication software, and can interoperate on a specific file basis with external software. Cadre has been prominent in fostering effort under the CDIF (CASE Design Interchange Format) standard committee to develop a CASE tool interchange standard.

Traceability

Teamwork has a new requirements tracing tool termed Teamwork/RQT which has not been evaluated by IDA. Teamwork/RQT was originally developed by SAIC under the name of THOR. The Status Label on diagrams and the Annotate feature can be used to capture some pertinent information. A general query on either would have to be done by a user-written search program.

Information Modeling Support

Teamwork supports the description of a system's data items and structures through either ERA or structure diagrams. Their content is captured in Teamwork's database and could be queried or analyzed by user-written programs via the ACCESS interface. No database normalization tool is provided. Further, the Teamwork database is not user extendible and does not support general user perceptions of the system under design, such as physical attributes or relationships among objects that are not central to one of its various editors.

Distinguishing Capabilities

Teamwork is a well-established dataflow diagramming tool with an easily-used interface. It is noteworthy that extensions are being made to increase its applicability for large defense projects. Key examples of this are the Ada tools and the Model Configuration Management facility.

APPENDIX C

TAGS

TAGS (Technology for the Automated Generation of Systems) a product of Teledyne Brown Engineering, Huntsville, Alabama. TAGS allows the user to define, analyze, and simulate a new system design. Its underlying design methodology might be called "Engineering Block Diagrams." TAGS provides certain diagrammatic and tabular forms to define and decompose a system's functionality down to a modular level. At the lowest level, modules are represented in a flow-chart form that embodies algorithms, synchronization, timing delays, etc.

The TAGS system is implemented for Apollo Domain workstations under the Aegis Operating System, Sun series 3 workstation family using Unix, and the DEC Vaxstation 2000 using ULTRIX. Porting also is underway to the IBM's version of Unix, the AIX Operating System. A VHDL compiler is underway, using TAGS source descriptions as input, and Teledyne Brown expects its completion in early 1990.

Teledyne Brown also offers the Requirements Verification Tool Set (RVTS) package, now implemented on IBM PC compatible hardware under DOS, but being ported to X-Windows Version 11 and ULTRIX. RVTS may be used with TAGS as a computer aid to capture free text requirements statements and to extract requirements based on key word phrases selected by the user. A fully integrated RVTS/TAGS version is scheduled for delivery soon.

The major components of TAGS are:

a. Input/Output Requirements Language (IORL)

b. Diagnostic Analyzer (DA)

c. Simulation system, with the Simulation Compiler

d. Automated Configuration Management (CM)

e. Executable Ada Code Generator (ECG)

f. Requirements Validation Tool Suite (RVTS)

Graphics and Editing

No significant change to the TAGS graphic interface has occurred in its recent enhancements. The Sun version is a close replica of the look and feel of the earlier

Apollo implementation. All TAGS versions are intended to have identical menus and icons, but some augmentation may take advantage of particular host features. For example, the Sun version already uses X-Windows and may be considered somewhat more user friendly than the Apollo version.

Design Semantics and Support

TAGS, as noted in the prior IDA evaluation, provides capability to specify systems in depth, using its various tables and diagrams. This depth is the basis for automatic Ada code generation, which is now available for simulation, prototype, or limited production purposes. The Ada source code may be compiled by the DEC, Alsys, or Verdix Ada compilers (or, presumably, any validated MIL-STD-1815 compiler) for any of their target environments. Teledyne Brown considers the source code to be of production quality, suitable for direct incorporation in operational code, but IDA did not evaluate this claim.

Team Design Support

The tool supports integrated work by multiple analysts/work-stations on a local area network, as explained in IDA's previous evaluation. No changes in this capability have occurred.

The Configuration Management (CM) package provides a set of features for tracking a system design database. With it, a user may catalog Engineering Change Proposals (ECP) and retain a historical record, including approved ones as Specification Change Notices (SCN). Version identification is available, and time-date stamping is used by the diagnostic analyzer. The prior IDA report has a complete summary; no enhancements have been released in the past year.

Documentation and Output

TAGS can produce Postscript formatted pages, which can then be read into external tools that accept Postscript files. Predefined interfaces are available to Interleaf and Mentor Graphic's Context publishing software.

Static Diagnostics

The TAGS Diagnostic Analyzer (DA) performs automatic checkout and validation of a design as described in IDA's earlier report. Once a design is validated, executable Ada code may be generated for a target environment, as stated above, by the Executable Ada Code Generator ECG).

Simulation

The simulation compiler (SC) produces an executable system model as an Ada program. During this step, some further static errors may be found. There are two ways of generating and running a simulation. In both, a statically correct TAGS definition is input to the simulation compiler and some part or the whole system selected for simulation. This step is done on the DEC, SUN, or Apollo workstation on which the definition was created. The SC output code is then handled by one of two alternatives:

a. The SC output Ada code is exported to a VAX VMS system and compiled with the VAX Ada compiler in conjunction with the TAGS simulation library and executive resident on VAX. The executable simulation is run on the VAX.

b. The Apollo workstation's Alsys or Verdix Ada compiler compiles the SC output, which is run on the Apollo workstation with the TAGS simulation executive and library.

In both cases, the simulation runs with the input conditions specified in the TAGS definition of the system. Thus, the input may be special simulation input files defined as input to a Schematic Block Diagram (SBD); alternatively, the input may be values defined in a Predefined Process Diagram (PPD) appearing as the final decomposition of some SBD.

Adaptability

TAGS adaptability features remain unchanged since the last report.

Interoperability

Library routines can be accessed with user-defined C and FORTRAN programs to import data from and export data to the TAGS database.

Traceability

The RVTS requirements list uniquely numbers the requirements and is stored in a relational database as the foundation for traceability and verification/validation or testing. Connecting numbers may be transferred manually onto TAGS design diagrams.

Information Modeling Support

Although the user can specify input and output record formats, there is no specific capability to model the information for database or other purposes.

Distinguishing Capabilities

TAGS is well engineered and continues to evolve towards capturing as much as practical of the system and software engineering process. TAGS facilitates the definition and simulation of a system as well as automatic code generation for target machines. Teledyne Brown states that they are working on a VHDL generation capability for producing VHSIC designs, consequently achieving an integrated hardware and software design system. Configuration management and traceability tools already are available, and being improved in new versions.

APPENDIX D

AUTO-G

Auto-G, a toolset developed by Advanced System Architectures (ASA) of the United Kingdom, was highlighted in the previous IDA report [Fife 87] for the scope and rigor of its graphical specification approach. ASA is now a wholly owned subsidiary of RJO Enterprises, Inc, which is based in Lanham, Maryland. Auto-G is hosted on Sun workstations, VAX systems via conventional terminals, DEC Vaxstation 2000/3100 workstations, Apollo workstations, and Atari personal computers. Aside from its specialized utility programs, such as plot generators, the toolset has these major components:

Component	Capability
Auto-G	comprised of the graphic editor and underlying database
Sema	the semantic analyzer or diagnostic facility
Sadmt	the translator from specification language to SADMT
Dbutil	design file manager
T-print	translates from the graphical (G) to textual (T) representation
T-parse	translates from the textual (T) to graphical (G) representation

Graphics and Editing

The Auto-G toolset incorporates the G graphical specification language and an equivalent textual language, T. Its menu, keyboard, and mouse interface is easily used, as IDA reported earlier. Its pick-and-drag panning remains an annoyance that is sometimes important because of the size of the "tree" comprising a G system description. This annoyance is somewhat alleviated through the use of the "Locate World View" command which allows any point in a design to be accessed by highlighting the desired point.

Design Semantics and Support

G specification capabilities apparently are complete regarding logical behavior and performance aspects essential to SDI. For example, a recent ASA application of Auto-G demonstrated a fully defined sensor data fusion architecture for SDI. The

specification approach proceeds from identification of concurrent processes and their interactions through interchanged messages. A process can be specified in enough mathematical and parametric detail to automatically generate compilable code representing its operation. Code generation has been demonstrated by ASA for SADMT (version 1.5), Ada, and C. Replication of processes, data structures, arithmetic processing, procedure calls, addressed message passing, and elapsed time or time constraints are among the aspects covered by G. However, physical components or allocation of resources to logical operations are not explicitly representable. Depending on their exact nature, they may or may not be representable indirectly or implicitly.

Team Design Support

ASA has added a capability to access logical Auto-G directories so that users can access design databases in other paths. Auto-G currently operates with Change Configuration Control (CCC) to provide software configuration management. Supporting the tasks involved here depends on operating system support for file access and time-date stamping. Auto-G's extensive versioning and view capabilities are helpful.

Documentation and Output

No change in capability here has been reported. Plotting distinct views is the primary way of getting hard copy for the G specification. ASA has indicated that it is currently implementing automatic generation of required documentation.

Static Diagnostics

Auto-G has a unique and effective technique of placing diagnostic messages on the G graph where each error applies. Diagnostic capability is extensive, commensurate with the specification depth that G provides. The prior IDA report [Fife 87] provides more information. No change in this capability has occurred.

Simulation

ASA has enhanced Auto-G's automatic SADMT generator for consistency with the IDA-released version 1.5 and the accompanying SDI Simulation Framework [Cohen 87].

Adaptability

No enhancements have occurred; see prior IDA report [Fife 87].

Interoperability

Auto-G can be complemented in limited ways by other Unix-based tools which will deal with files irrespective of their content, such as the Source Code Control System. No major tools other than those from ASA have been closely integrated with it. Design information can be imported or exported in the form of ASCII coded, T language statements.

Traceability

Auto-G has no feature explicitly aimed at the traceability requirement. Extensive comments can be placed on a specification diagram, and specific instances located later by a query and highlighting technique.

Information Modeling Support

All data item or structure definitions can be dumped to a file for external processing, and a basic query function permits locating instances on the G diagram through a highlighting technique. The specification language covers all data type characteristics conventionally found in programming languages, including templates and replication. It has no high level, predefined representation for database access, nor a general dictionary facility for textually describing the meaning or semantics of data. A new data dictionary program called Datadic is expected soon to provide a selective data dictionary query facility.

Distinguishing Capabilities

Auto-G is especially noteworthy for the rigor and completeness of its specification capability, and the extent of its development toward automatic code generation. It is the only CASE tool which provides automated support of SADMT for SDI simulation needs. Its user interface is easy to learn and use, and reflects up-to-date techniques except in scrolling or panning. It is primarily limited by its lack of supplementary features for large project applications, such as documentation and configuration management support.

APPENDIX E

DCDS

DCDS/Ada is a tool developed by TRW in Huntsville, Alabama, under contract with the U.S. Army Strategic Defense Command (USASDC). All rights to DCDS/Ada software and documentation are owned by the U.S. Government. DCDS/Ada is distributed by direction of the USASDC in Huntsville, Alabama, is subject to export restrictions, and may not be further distributed to any domestic or foreign institution, organization, or individual. DCDS/Ada has behind it many years of development and use on defense projects, dating back to the original 1973 SREM research.

DCDS/Ada release 4.4 is currently available for VAX/VMS platforms. DCDS/Ada release 4.3 is available for the Sun workstations. The IBM-AT has been dropped as a supported platform for DCDS.

DCDS/Ada release 4.0 was released in October 1988. This was a major release which added improved hierarchy capabilities, simulation for RSL (the software requirements analysis language), a stand-alone abbreviated simulation tool, automated configuration management within DCDS, automated document control, limited support for the spiral model methodology, and support for the SDI System Description Language (SDL).

Graphic and Editing

The Sun version of DCDS has a greatly improved user interface and graphics editing capability over the version evaluated by IDA in 1987. The Sun version of DCDS takes advantage of the Sun's mouse, windowing, and graphical capabilities. Some operations, such as entering a line of text in Browse, overuse the windowing capabilities, slowing user entry. Data flow between functions on F_nets is still not visible.

Design Semantics and Support

DCDS/Ada embodies five different design methods with a corresponding language for each method:

Method	Language	Purpose of Method
SYSREM	SSL	System Requirements Engineering Method for defining systems and their behavior.
SREM	RSL	Software Requirements Engineering Method for defining the requirements of software.
DDM	DDL	Distributed Design Method for defining architectures consisting of hardware and software.
MDM	MDL	Module Development Method for defining units of program code.
TSM	TSL	Test Support Method for testing.

Each language consists of an Entity-Relationship-Attribute (ERA) database schema with various constraints. Constraints on the schema are checked using the DCDS Query facility and check files which have been developed by TRW.

DCDS/Ada provides for timing specifications via "validation points" in RSL, as well as providing for the textual attribute "performance_index" in SSL. The notations embedded in DCDS/Ada can describe hardware components as well as software components and the describe the allocation of software to hardware. DCDS/Ada can formally describe process behavior, including concurrent and replicated behavior.

Team Design Support

DCDS/Ada allows for multiple designs to be included in the database, each of which is called a "configuration". There is no support for automatic splitting and recombining of the database, nor for sharing the database between multiple simultaneous users.

DCDS still has difficulties if a user decides to make a change in a language previous to the current language - all information that was added to a later design language is lost when moving data. Each DCDS language has its own separate database. Each transition to the next phase of design requires that a translation program be run which creates the next phase's database with some entities from the previous phase's database. A change in a previous phase's database requires a re-translation into the database of the next phase, erasing all of the added information in the following database. For example, if the user decides to add information to SSL, which is used to describe system behavior, after entering data to RSL, which is used to describe software, all data in the current RSL database will be lost when the SSL data is transferred to an RSL database. This transition can be done manually to avoid this problem, but with difficulty, thus creating the potential for masking subtle errors. This is a serious deficiency for use in large scale systems.

Documentation and Output

The DCDS documentation has improved over previous versions but is somewhat dated. Most of the documentation currently available is dated October 1987. The Methodology Guide Appendices (CDRL A008) omit the comments on each entity, relationship, attribute, and qualifier. These comments are necessary for complete understanding of the schema. These comments are available on-line.

DCDS/Ada has an automated document generation capability for creating text files for four of the DoD-STD-2167A documentation types. These are the SSS (System Segment Spec), SSDD (System Segment Design Document), SRS (Software Requirement Specification), and SDD (Software Design Document). Graphics cannot be included in the output document, and the output is insufficiently formatted that the resulting text file will need to be edited before it can be printed.

The "report" command was added to the Query language in DCDS version 4.0. This command may be used for report generation. Report uses a template to generate information about various objects; these templates may be developed on the VAX using a template generation tool but must currently be created manually on the Sun. Templates created on the Vax may be used by the Sun version of DCDS.

Static Diagnostics

DCDS/Ada contains a large set of static diagnostic tests. New static diagnostics can be easily added or standard ones modified since all static tests are stored as ASCII files which are sent to the Query facility. The Query facility has been improved by adding constructs for searching hierarchies, partial hierarchy listings, and creating sets using hierarchy definitions. The flexibility of the output display has been improved by adding the MAP, FULL_MAP and GROUP options to Query.

Simulation

The first two languages, SSL and RSL, have simulators available. There is also a stand-alone Abbreviated Simulation Tool (AST) for building and executing small simulations. There are no tools for dynamic validation nor for generating a self-standing simulation program.

Adaptability

The underlying ERA database system is very flexible. The tool supports modifications to existing schemas and the construction of a new ERA database schema. Queries and reports may be generated with the new schema. The graphical editors may not be adapted.

Interoperability

DCDS has been augmented to produce SDL diagrams. It is unclear if these diagrams could be fed into other CASE tools which support SDL. All data may be moved in and out of DCDS as text files.

Traceability

DCDS/Ada provides complete traceability back to the system requirements. DCDS/Ada can record decisions made, who made those decisions and when these decisions were made. DCDS databases cannot directly trace to other databases, so copies of the entities of other databases are included in the current database for this traceability to be available. Allocations of one object to another can be represented.

Information Modeling Support

DCDS/Ada is based on five separate ERA database schemas which are completely visible. Entities, relations, attributes, qualifiers (attributes on relationships), synonyms, and reverse relationships are all visible and modifiable by the user. The schema definitions can be queried on-line. Inheritance is not supported.

Distinguishing Capabilities

The design schema of the ERA database and its checking facilities are especially easy to modify.

APPENDIX F

RDD

RDD-100 (Requirements Driven Development) System Designer, called RDD-100 in this paper, is a computer-aided systems engineering (CASE) tool developed by Ascent Logic Corporation, San Jose, California. RDD-100's concepts are based upon the first of the five stages of the Distributed Computing Design System, DCDS. Thus RDD-100 supports the System Requirements Engineering Method (SYSREM) and its System Specification Language (SSL). RDD-100 differs from the current implementation of DCDS's SYSREM support in that it has an improved user interface, including a superior graphical interface. RDD executes on a Sun/3, Sun/4, Apollo DN-3000 or 4000, or Apple Macintosh II, and requires 8 Mbytes of RAM on any of those platforms. Ascent Logic plans to produce a series of products which use concepts similar to DCDS; this expected series of products will be referred to as the RDD family, differentiating this series from the currently available tool, RDD-100.

Graphics and Editing

RDD-100 is, in general, significantly easier to use than DCDS. Editing is easier to perform, and the product takes advantage of the windowing environment. Text can be cut or copied from one window and pasted into another. The graphical editor has a relatively quick response.

One of the more striking aspects of RDD-100's graphical editor is its "expand in place" function. This allows the user to see decompositions of a function in the place of the function as well as the higher-level functions simultaneously. This view of multiple levels of decomposition may also be printed on a Postscript-supporting printer. Expanding in place is a solution to the problem of many tools' not being able to display both the higher and lower levels of details simultaneously.

The graphical editor also has the ability to turn on and off some of the display aspects of function networks (F_nets), thus reducing visual clutter. The editor can turn on and off the display of items (data structures) or item flow, thus making it possible to see the functions and their order of execution without having to see the data flows accompanying the functions.

The graphical editor automatically lays out item flow but cannot, unfortunately, do so in a "pretty" fashion. Item flows are drawn as a straight line from an item to a function no matter where the function is, and the results may be misleading since any function

that is between the item and the function may appear to be connected. The tool does allow a user to improve the legibility of the F_nets by moving functions down from where they would normally be drawn. This moving of functions is accomplished by editing a text field of the functions in an F_net. These modifications to improve the legibility of the F_nets are somewhat clumsy; it would be better if the tool allowed the user to move a function directly with the mouse, and better still if the tool automatically laid out the F_net.

Design Semantics and Support

RDD-100 has essentially the same specification depth as SSL in DCDS. RDD-100 is concerned with system level design, with information as to the ordering of functions and allocation of functions to components (subsystems). As in DCDS SSL function behavior can be represented using the constructs of sequential, alternated, iterative, concurrent, or replicated execution.

Although RDD-100 has a number of constructs that can completely specify order of function, it is somewhat weak in specifying time of performance, as is required in real-time systems. Aggregate data structures, called "TimeItems," show the arrival rate of input or output, and aggregate functions, called "TimeFunctions," can specify fixed times of execution (for example, once every five seconds.) RDD-100 TimeFunctions also have an attribute "Performance Index" which may be used to describe timing characteristics of a system, including specifying maximum allowable time for performing a process in a real-time system. While a maximum time may be entered in a performance index, the timing requirement is not processed by the tool, and is not currently used for scheduling or other types of analysis.

RDD-100 is not meant to handle software design, but rather the higher level system design. Ascent Logic plans to later release a product for software design. The only concept RDD-100 has for describing hardware or software is the entity "component," which might be hardware, software, or even a contractor, but is most likely a subsystem. Thus the tool is designed to allocate functions to subsystems.

Team Design Support

RDD-100 has extensive support for splitting a design database into smaller databases and for recombining these databases under various conditions and user control. There is no support for sharing a database on-line between multiple simultaneous users. This is because the tool loads the entire database into memory before the database can be

used. Ascent Logic states that on-line database sharing will eventually be provided by products under development. Alternative designs may be stored in RDD-100.

Documentation and Output

The tool documentation is indexed and includes many sample diagrams. The materials are essentially reference material with brief introductions to the methods of RDD; the manuals do not include detailed descriptions of the methods and techniques supported by RDD, but instead suggest that the users first enroll in a training course by Ascent Logic.

A number of standard reports, such as those required by MIL-STD-2167A and MIL-STD-490, are included in the package. These reports may be modified for a particular type of user report. The report generator can generate graphs of F_nets and I_nets of varying detail with a PostScript printer and include them in a report. RDD-100 uses a report language similar to F_nets. This reduces the learning time for creating new reports as opposed to learning another language. A user must create or modify a program to produce a different type of report.

Ascent Logic provides software for generating IDEF-0, N2, and user definable hierarchy charts from the database.

Static Diagnostics

RDD-100 has a number of built-in checking phases which cannot be altered by the user. The report generator may be used to create additional diagnostics, but these reports will be produced at a much slower rate. The report generator can be used to create, modify, and delete elements.

Simulation

RDD-100 has no self-standing simulation program, but a user may transfer an SSL design to DCDS and use its simulation program. Ascent Logic is working on a simulation facility which it calls the Dynamic Validation Facility (DVF).

Adaptability

RDD includes an extension capability which allows users to add entities, relationships, and attributes to the existing schema and to even create new schemas. Changing the tool's graphic symbols, their semantics, or the built-in checking phases of the tool is not possible. Reports are easily changed and new ones are relatively easy to create once the user understands the report generation language.

Interoperability

Data from RDD-100 may be transferred to DCDS with a translation routine, though such a routine is not included with the tool. This allows systems described using RDD-100 to be simulated using DCDS's SSL simulation program.

Traceability

RDD-100 provides complete traceability back to system requirements in the original system specification documents, the decisions that were made, and the engineer who specified the information. RDD-100 can also specify the allocation of components (generally subsystems) to system functions.

Information Modeling Support

RDD-100 and DCDS have a very similar conceptual model which forms a system design information model with predetermined entity types, attributes, and named relationships. The underlying model partially supports inheritance. Inheritance of attributes and relationships is supported by the ERA extender of the tool, but relationships are not allowed to abstract (ancestor) entities and these abstract entities are not visible outside of the extender. Every system entity has a "traced from" relationship to describe the source of system information. This model is available as two charts, a schema chart and an attribute chart, which are included in the reference guide. The entities are also listed on a menu of the editor, and the relations are visible when expanding each entity. Thus this model is clearly visible to the user. RDD-100's extension capability, when available, will allow users to add to the model for important systems aspects of their particular type of system.

Distinguishing Capabilities

RDD-100's graphical interface is easier to use than DCDS. Its report generation capabilities are superior to DCDS and includes IDEF0 graph generation. The RDD-100 schema only represents the first of the five stages of DCDS. Like DCDS, RDD-100's F-nets show both functions and behavior on the same graph. RDD-100 can show allocation of components (hardware or software) to system functions and can record traceability back to system requirements in the original system specification documents. Since RDD-100 only covers the first stage of DCDS, it does not cover software design or software to hardware allocation. A system designer using RDD-100 must switch eventually to a software design tool. One option is to switch to DCDS for areas RDD-100 does not cover, since DCDS and RDD are very similar semantically.

Ascent currently intends that its products will not be split into multiple databases as the DCDS databases are currently implemented. The multiple databases of DCDS make it difficult to keep data consistent. DCDS users need to use ad hoc data recovery methods after a change in a higher-level database because any change to information in a previous database causes all information in subsequent databases to be lost. Since these future phases of RDD are not yet available, it is unclear if RDD will be able to successfully avoid this problem. Current users of RDD who switch to DCDS for the remain portion of a system design proces will still be troubled by this difficulty.

APPENDIX G

STATEMATE

STATEMATE consists of a set of modules developed by i-Logix of Burlington, Mass. Its underlying design methodology has been presented in several published papers in the open computing literature [Harel 87, Harel 88, Harel 90]. STATEMATE provides a dataflow diagram (called an Activity Chart), and behavioral specification through a state transition diagram (called a State Chart). Both diagrams may be decomposed. Physical allocation of software to hardware is also definable through one or more Module Charts. Activity and State Charts allow specification of concurrent processes. State Charts are connected to Activity Charts through control flows specified on the dataflow diagrams, and define activations of the processing steps shown on the Activity Chart. State Chart decomposition establishes a more practical basis for using state transition specifications for complex systems.

The STATEMATE system incorporates the InterBase 2.0 (or higher) DBMS with its delivered package. It is implemented for three platforms: Sun series 3, with SunOS 3.4 and NeWS 1.1 software; DEC VAXstation GPX or 2000, and MicroVMS and UIS software; Apollo Series 3000 or 4000, with Aegis, Domain IX, and GMR graphics software. The next release of STATEMATE will support the ULTRIX and VMS operating systems, DEC Windows, X-Windows, and RISC-based Sun and DEC workstations. X-Windows

STATEMATE has four major and separately purchasable components:

a. Kernel, containing the three graphic editors;

b. Documenter, providing customized output, e.g., for MIL-STD-2167A;

c. Analyzer, producing a visual simulation of the system; and

d. Prototyper, producing Ada code to run on a target machine.

Graphics and Editing (Ease of use)

The use of graphics and the tailorability of the medium are excellent. As an example, a color may be used to identify a mechanism or concept and may be changed easily to a different color. The windows and pull down menus form an effective user interface, and the commands are relatively easy to learn. However, the symbols (icons) are all rectangles with certain differences, but a very similar appearance. A non-color

terminal may prove difficult to use due to the lack of shape distinction of the different icons.

Design Semantics and Support (Specification Depth)

State Charts are an extension of state transition diagrams which add concurrency and hierarchical decomposition.

The approach of defining processes by dataflow and State Charts is clear and effective. The system allows formal definition of sequential, iterative, replicated, or concurrent processing. Specification of iteration, though possible, is difficult to implement, requiring individual states for each iteration. Replication of elements (in the sense of an index on a logical or physical object) is not yet available, though the vendor states that the company is working on adding this feature. Timing is specifiable, in the sense that it is possible to restrict the flows in parallel (as in the Ada rendezvous) but the overall time cannot be constrained (i.e., any overall timing constraint is only observable or checked at the simulation level). Dynamic allocation of a particular function to specific hardware (during a special phase of the processing) is not possible. It is not possible, from the functional charts, to see what hardware is to run which functions, though the Module Charts have this information.

Team Design Support

STATEMATE provides for multi-user design of systems. Each user's work is maintained in a local work environment. When a design is baselined, the design data is written to the databank, which is a common area (i.e., central database). Design data is stored in ASCII document form in the databank. STATEMATE performs logging as well as versioning of files.

In order to work with large designs, users can move in and out of editors as they move through various parts or decompositions of the system under design. A tool is also provided to facilitate the accessing of graphical representations.

Documentation and Output

Design data and other information contained in the STATEMATE database can be output on printers or plotters. STATEMATE's Graphic Kernel is a query mechanism for retrieving designs, documentation, and other information from the STATEMATE database. This feature can be used in conjunction with the MIL-STD-2167A documentation template to automatically generate MIL-STD-2167A documentation. Reports may be generated and the format stored for about 12 different forms for future use. The user can also define templates for report generation. STATEMATE supports Postscript, troff,

and nroff formats. In addtion, the databank stores all design data as readable ASCII text. STATEMATE's DATAPORT consists of a set of C routines that the user can use to build a translator to import and export ASCII data from the database. Built-in interfaces exist for publishing tools such as Interleaf and FrameMaker.

Static Diagnostics

Basic testing includes validating flows, detecting orphan activity, and checking completeness of activities. The function/module allocation can also be checked for completeness.

Simulation

The Analyzer may be used to initiate a simulation of a design. It first checks the reachability of all states, then checks for deadlocks. The simulator also helps to detect race conditions. Unused transactions and non-deterministic behavior are detected and flagged. Simulation is somewhat coarse relative to hardware resources, since it addresses only the activation or deactivation of machines at any instant. Although a demonstration to IDA showed simulation with the dataflow and state transition diagrams visible, large problems may not be sufficiently decomposable to permit both diagrams to be visible during simulation.

STATEMATE generates Ada and C code. Ada language generation interfaces can include graphic panel routines for special outputs. The Ada and C code that is automatically generated is fully compilable. I-Logix is currently extending its tool to provide generation of VHDL.

Adaptability

The system does not allow tailoring of its icons, nor direct access or change to the underlying database (e.g., the schema cannot be augmented by the user). The schema and file details can be obtained under special license. Library functions can then be accessed.

Interoperability

All diagrams in the file may be dumped into an ASCII form for retention or transfer.

Traceability

STATEMATE supports traceability between design elements and STATEMATE forms. Forms can be either formal textual information (i.e., requirements or other information stated in mathematics or logic notation), non-formal notation (e.g., free text), or

attribute fields. If free text is used, the user must specify where the text resides by entering the directory path. Attribute fields can be used to define a classification system which can then be used along with STATEMATE's query facility to trace requirements and other information to design elements. Elements may be annotated for traceability, but there is no current plan to link these comments with other software or tools.

Information Modeling

Textual data definition forms are the means of describing a system's data items and structures. No graphical data structuring tool, such as an Entity, Relation, Attribute diagrammer, is incorporated. There is no specific capability to model information for database design or other purposes. STATEMATE does have a query and reporting capability of its data.

Distinguishing Capabilities

Distinctive features of STATEMATE are its State Charts, Ada code generation, and animating simulation of the defined system. State Charts extend state transition diagrams to add concurrency and hierarchical decomposition. STATEMATE's specification of events, conditions, and actions allows timeouts and dependency on other states to be expressed.

APPENDIX H

REFINE

Refine is a product from Reasoning Systems, Inc., of Palo Alto, California. Refine includes four major sections:

a. A very high-level, wide-spectrum, executable specification language;

b. An object-oriented database for specification objects and documents;

c. Tools for generating Common LISP code from specifications; and

d. Tools for customizing Refine for maintaining and re-engineering existing software.

The specification language may be considered very high-level because it includes declarative, set-theoretic and first order logic language constructs such as sets, mappings, and quantifiers. The specification language may also be considered wide-spectrum because low-level constructs written in Common LISP can be freely intermixed in the specification text and because a variety of programming paradigms, including procedural, object-oriented, and transformational programming are supported.

Refine runs on Sun-3, Sun-4, and Sun SPARC workstations, Hewlett-Packard Series 300 workstations, Texas Instruments Explorer and Micro-Explorer, and the Symbolics workstation.

Graphics and Editing

Refine runs under the X11 Window system but all its specifications are entered textually. GNU Emacs, a common reconfigurable text editor, may be used to enter Refine specifications. Refine includes a menu-based knowledge base browser called BROWSE which displays an object and allows the user to traverse across the database to see the attributes of an object and its relationships to other objects. Refine also includes a menu-based knowledge base editor called EDIT-OBJECT that is similar to BROWSE but which allows the user to edit an object's properties. Online help is available.

Reasoning Systems has implemented a toolkit, called the User Interface Toolkit, for creating interactive graphics tools. This has been used, for example, to create tools which accept state transition diagram specifications for communication protocols, and tools which graph the structure and dataflow properties of C, COBOL, and JCL

software. This tool was not available at the time of the review but is now available in Refine 3.0. With this tool the user can assign arbitrary meanings to icons, associate icons with particular object classes, and create relationships by linking objects together. Diagrams entered by the user have a corresponding textual description generated automatically.

Design Semantics and Support

It is difficult to evaluate the design semantics for this product. If viewed and used as a programming language, Refine does not directly support software to hardware allocation, concurrency or replication. Maximum timing might be specified using a LISP construct. Refine does not support concurrent or replicated execution directly, although many of its high-level constructs do not express order and a different compiler could perhaps implement some of the constructs as concurrent or replicated processes. If extended by the user and used as a knowledge base with a specification language, such concepts as timing, allocation, and concurrent execution could be represented in the knowledge base, but these concepts are not built into the basic Refine system.

A different approach for using the product is as a basis for creating application-specific objects or languages which could then be used to design a system for that application. The application-specific information could include specialized languages, object classes, and tests. This application-specific information and knowledge base could include information about timing, allocation, concurrency and replication, and could specify meaningful communications primitives for the application.

Team Design Support

The Refine knowledge base has a facility called a "context mechanism", which allows competing designs to be generated and analyzed. This facility also makes it possible to save the state of a knowledge base and to later restore the knowledge base to its previous state. The knowledge base has a facility called the Persistent Knowledge Base (PKB) which allows the user to designate an area of the knowledge base that can be saved and shared by other users. Simultaneous users are allowed to access this shared knowledge base, and are warned but not prevented from accessing the information simultaneously. If both users change the knowledge base there will be two versions which are branches of the original knowledge base; unfortunately, there is no merging facility for recombining the two versions of the knowledge base.

Documentation and Output

The documentation does not have separate tutorial and reference sections, but is indexed, ordered by information category, and includes many examples. Standard reports such as MIL-STD-2167A are not included. The knowledge base may be queried and request for objects to display themselves in a meaningful way but there is no specialized report generator tool to make report generation easier. Reports may be generated by creating them in the Refine specification language. Refine can generate Common LISP code which can then be ported and compiled. Reasoning Systems is also working on a project to generate C code from the Refine specification language. Applications which utilize the knowledge base but do not require rerunning the compiler may be executed without Refine but with a runtime support package called RERUN. Applications which require the specification compiler to execute require the full Refine package. For example, an application which is a specialized CASE tool for a specific problem domain which accepts user diagrams, generates Refine specifications, and then creates the Common LISP code from the specifications would require the full Refine package. The resulting code from this application could be compiled by a Common LISP compiler and run without Refine or Rerun.

Static Diagnostics

The Refine specification language compiler checks for syntax violations, as does the syntax checker of the program transformation tools for interactive use of both the Refine specification language and for other languages. The specification language compiler also warns of dead code. Static diagnostics that can be expressed as rules and patterns could be easily created using the Refine specification language. For communication protocols Reasoning Systems has implemented tests for deadlock, livelock, unreachable, and unused states.

Simulation

Refine generates Common Lisp code which can be executed and/or ported to other machines. Assertions can be included using the specification language which are then tested at execution time.

Adaptability

The Refine object-oriented knowledge base is extensible, as are the syntactical transformation tools. The User Interface Toolkit allows the user to adapt semantic meaning to graphical icons. Refine contains tools to adapt its database to capture, analyze and transform software written in other languages. Users have used Refine to handle software written in Ada, C, COBOL, FORTRAN, and other languages.

Interoperability

Refine is currently being used by various users as a back end for other CASE tools. Various users have translated Software through Pictures data, statecharts, and Petri nets into Refine and then used Refine to produce code.

Traceability

The Refine object-oriented knowledge base contains information about relations between requirements, specifications, programs, documentation, test suites and test results. The tool can propagate automatically information from some objects to another. Information is stored in the knowledge base regarding who performed what operations on what objects. This information can be queried. While there is no built-in 'reason for decision' object, an object schema for storing decisions and the reason for making those decisions could be created in a matter of minutes.

Information Modeling Support

The Refine knowledge base is object-oriented, and the information model is completely visible. The knowledge base is based on three central concepts: object classes, objects, and attributes. Attributes may be used as maps to other objects, thus becoming relationships. Reverse relationships, called "functional converses," are directly supported by the knowledge base. Attributes may be defined to be computed on demand. Single inheritance is supported. Querying can be performed on the schema as well as on the instances using the Refine specification language. Object name synonyms, relationships on relationships, and attributes are supported. Refine does not include multiple inheritance nor a built-in natural language query language. A natural language query language could be created using the parser generation system.

Distinguishing Capabilities

The Refine specification language is unusual in scope, and sets Refine apart from other tools. Its intent is to provide a very high level, mathematically formulated specification language which can be translated into a common computer language. The language includes some first-order constructs of predicate calculus such as for-all ($\forall$), there-exists ($\exists$), implies ($\rightarrow$), and Horn clauses, the latter as rules. It includes set theoretic data types such as sets, mappings, and sequences, an extended one-way unification and pattern instantiation facility, rules, and constraints. For object-oriented programming it includes object data types with specialization and inheritance. The specification language also includes standard structured programming constructs such as "if" and "while". The language is strongly typed and can directly include Common LISP programs. Refine's specification language can use standard infix mathematical notation

unlike some other computer languages which can reference LISP.

One construct of particular interest in the Refine specification language is the 'transform' construct. This allows the user to specify certain kinds of state transformations by specifying the pre-condition and post-condition of the transform. The sequence of actions to actually produce this transform is then determined by the Refine compiler. The compiler can determine which conjuncts in the antecedent are enumerated and which are tested, the nesting order of enumerations, and the order and manner conjuncts are made true.

Another unusual aspect of Refine is its integrated object-oriented knowledge base. The knowledge base models objects and relationships and makes it easy to add application-specific objects and object classes. This object base is used for storing software and software-related objects (such as documentation, test cases, bug reports, and so on). As mentioned in the information modeling section, the knowledge base is based on three central concepts: object classes, objects, and attributes. Attributes may be used as maps to other objects, thus becoming relationships. Reverse relationships, called "functional converses", are directly supported by the knowledge base. Attributes may be defined to be computed on demand. Single inheritance is supported.

APPENDIX I

SPECTRUM

Spectrum is a system under development by Software AE, Rossyln, VA. It is hosted on Sun workstations. It provides an information modeling or knowledge-based approach to design specification, and automatic generation of prototype code to implement the specified application. Under contractual support, it has been used in prototyping an Acquisition Manager's Assistant for naval weapon system acquisitions.

Graphics and Editing

Spectrum is used with a predefined set of windows which address various elements of its conceptual model and operation. Specifications are rendered as text entries, not by graphic icons. Particular windows become available at different levels of a specification tree that guides a designer in fulfilling specification requirements. Pull down menus are available to make selections or inquiries pertinent to the subject of a window.

Design Semantics and Support

Spectrum provides a semantic data model as a design schema. This is broken into two distinct parts, called the "World Model," which addresses the entities, relations, events, and actions in the application domain, and the "External Interface," which addresses the distinct display screens, menus, keys, and paraphernalia of interaction with the executing target code.

Team Design Support

No capabilities explicitly for team design support have been built into Spectrum as yet.

Documentation and Output

The specifications created for the World Model and External Interface can be printed from system files. The significant ouput of Spectrum is automatically generated code, in the C language, to implement the specified application. The code generation process is based on templates that are internal to Spectrum and that in part are used in Spectrum's implementation as well.

Static Diagnostics

The specification tree, windows, and menus limit a designer to just those choices that are potentially valid at any point. There are no other diagnostic capabilities.

Simulation

None. Spectrum rests in part on a conception that testing requirements are substantially reduced by automatic code generation from pretested templates. Hence, it may be argued that running the prototype system provides the most concrete measure of its potential performance.

Adaptability

The World Model and External Interface are general purpose. No adaptation of the windows, specification tree, or models is possible.

Interoperability

No particular cases have been explored for interoperation with other tools, so candidates, benefits, and constraints are unknown.

Traceability

No explicit means of tracing specifications back to requirements is provided.

Information Modeling Support

Spectrum is based on a predetermined design information model.

Distinguishing Capabilities

Spectrum represents an information-oriented design approach that presumably is suitable for application specialists without strong ADP/programming background. This approach is expected to enhance potential reusability of specifications and code, but no experience has been gained in that area. Spectrum's particular semantic data model is unique. Spectrum has been applied to specify an Acquisition Manager's Assistant, for use in Navy weapon system procurements.

APPENDIX J

DESIGN/IDEF AND DESIGN/FAMILY

The phrase "Design/family" refers to the set of products developed by Meta Software Corporation of Cambridge, Massachusetts. This set includes Design/IDEF, Design/CPN, MetaDesign, and Design/OA. Design/OA is a set of library routines for creating graphics and processing text which is used by all of the other tools. MetaDesign is a general-purpose drawing tool. Design/IDEF is a drawing tool specially designed for creating and editing IDEF-0, IDEF-1, IDEF-1X, and E-R diagrams. Design/IDEF can also perform some simple static checks. Design/CPN is a tool that can simulate annotated IDEF-0 diagrams from the Design/IDEF tool by converting the IDEF-0 diagrams into a hierarchical colored Petri net (HCPN) and simulating the resulting net. Design/CPN can also simulate HCPNs directly. Most of this review concentrates on Design/IDEF combined with Design/CPN.

Both Design/OA and MetaDesign execute on the IBM-PC and the Apple Macintosh computers. Design/IDEF executes on a Macintosh with at least 2 Mbytes and the IBM-PC. Design/CPN executes on a Macintosh II or on a networked configuration of a Macintosh and a Sun workstation. A version of Design/family under X-Windows for the Sun and DEC workstations is being created now.

Graphics and Editing

The tools MetaDesign, Design/IDEF, and Design/CPN have an effective graphical interface. Objects can be laid out in any manner and manipulated via a mouse. Design/IDEF does not have an expand-in-place display option as do RDD and STATEMATE. This means that diagram decompositions cannot be seen in the same diagram as higher diagram levels, but must be shown in separate windows.

Design Semantics and Support

The semantics of Design/IDEF are determined by annotating IDEF-0 diagrams and translating the annotated IDEF-0 diagrams into HCPNs. HCPNs are an extension to Petri nets which add "colors," roughly analogous to types in computer languages, and hierarchy, which is similar to the hierarchical layout of many graphical notations [Chiola 88, Jensen 81]. The HCPN notation is quite powerful, as evidenced by the large number of analysis tools which Meta Software plans to build in the future. Process behavior is described quite formally, but timing behavior is currently difficult to specify. It is unclear how easily the tool can describe software to hardware allocation. The Design/Family

does not fully support the relationships of data in the system being designed.

Many systems' successful performance depends upon the time the system or portions of it take to execute their intended function. There are various extensions to basic Petri nets that can be used to represent timing information such as, [Coolahan 83] but additions to HCPNs to represent timing which can be analyzed analytically are still an ongoing research topic [Marsan]. It is currently possible to create a "master clock" and put code into each transition of the simulation to model time, and/or add attributes in tokens to represent time, but this is not a simple approach, nor can this timing information be fully analyzed analytically by the tool. Thus HCPNs do not currently represent timing in a simple manner.

Mechanisms of the IDEF-0 diagrams are translated into tokens of an HCPN, and thus hardware for implementing software can be treated as a specially colored token. It is unclear if this representation can fully represent software to hardware allocation. Hardware to software allocation could also be noted as a comment. Investigating this aspect of the HCPN notation would require a more detailed report.

Process behavior is described formally in Design/CPN, and in fact a strength of Petri nets is the ability to describe concurrent activities requiring occasional synchronization. States, transitions, concurrency, and sequences of operations are easily represented with Petri nets. Replication can be represented in HCPNs as a color set with a subrange (essentially an index on a color set) or as different instances of a subnet.

These tools do not fully support the relationships of data in the system being designed. Design/IDEF can draw IDEF-1, IDEF-1X, and standard E-R diagrams, but does little with the resulting diagrams other than perform simple static checks.

Team Design Support

Design/IDEF includes functions for extracting and combining the database and models (the diagram set). The extraction routines create records for a subsection of the database, and the merge routines copy information from one database to another, typically using a master database as its destination. There is currently no automatic sharing of the database between multiple simultaneous users. There is no version identification or configuration management system.

Documentation and Output

MIL-STD-2167A reports are not automatically generated by Design/IDEF. Simple reports, such as a list of every box or of every arrow, are available in Design/IDEF. Executable code is not generated by the tools.

Static Diagnostics

Design/IDEF contains a few simple static diagnostics such as the presence of names on all arrows and boxes. Design/CPN checks for a number of additional annotations to the IDEF-0 diagrams that it requires, including arc expressions and guards, annotations on channels, and the data dictionary type information. Future directions include a number of more sophisticated diagnostics such as algorithmically determining deadlocks, livelocks, unsafeness, and unreachable places.

Meta Software Corporation plans to create an extended version of Design/CPN, termed Design/CPN Palette, which would add various tools for formal analysis. This would include construction of occurrence graphs (representing all reachable markings) calculation and interpretation of invariants, check of structural properties, and reductions (which shrink the net without changing selected properties). Occurrence graphs allow the user to determine how the system could enter specific states, perform simple statistics, and determine strongly connected components. Calculation of invariants would allow the user to specify conditions that should always be true, which the computer can then use to algorithmically determine if the condition can ever become false.

Simulation

Design/CPN can animate either the set of annotated IDEF-0 diagrams or an HCPN. Code segments in the general-purpose Standard ML computer language can be attached to the transitions of the simulation. These code segments could be used for such purposes as instrumentation, communication with other processes, and extension of the HCPN model.

Standard ML is a general-purpose computer language originally created at the University of Edinburgh, Scotland. ML is a functional language; this means that functions are first-class objects which may be passed as arguments, returned as results, and stored in variables. Like Ada, it is strongly typed and supports exception handling and abstract types. Like Common Lisp, ML is often used as an interactive interpretive language, and ML automatically determines the type of objects to satisfy its strong typing. Like both Ada and Common Lisp, it is statically scoped and has a module facility [Harper 86]. Although ML is not as common a computer language as Ada, Pascal, or LISP, it is a proven language.

A user can instrument a simulation in Design/CPN by writing ML code for animating the system diagrams, storing information in log files, performing statistics, or other instrumentation needs that could occur on a system transition. The animation of a diagram can be performed by changing shading, color, or line-thicknesses in the

diagrams. Since ML is a general-purpose language ML can also be used to represent information not easily modeled by HCPNs, such as changing attributes of the net tokens or determining what token should perform a transition (technically, what set of tokens should be bound). ML can also be used to communicate with other processes. The tools do not generate self-standing code.

Adaptability

It is possible to add comment icons and change the display attributes of icons in Design/IDEF. Design/IDEF does not have user-tailored semantics, and the underlying database schema is not available to the user. MetaDesign allows the user to choose various icons since it is essentially a drawing tool. The ML language and Design/OA library can be used to access and change diagrams. Source code is available.

Interoperability

There are no current products that interface to other CASE tools. There is a "reference report" in Design/IDEF that can output all data in the database, but the report does not include the schema used in the current database. There is currently no generalized import function, nor is there any function to import or export the database schemas. Design/IDEF can output Macintosh pic format.

Traceability

There is no built-in traceability to system requirements, for recording decisions made, or for who made what decisions when.

Information Modeling Support

The Design/family database is not relational, cannot be accessed by SQL, and does not support many information modeling concepts such as reverse relations or inheritance. Data storage and retrieval is split into two sections, a low-level set of routines used by tools and a high-level set for use by users. The low-level set is accessed using ISAM and is a doubly indexed B-tree, each record of which is indexed by record name and record type. Design/IDEF contains high-level access routines for creating record types, displaying records, and editing records. It limits record types to 7 fields per record, and each field can be a complex field pointing to another record type. This indirection can go a maximum of ten levels deep. The set of all record types and the data they contain is considered the database for a project. Diagrams are not stored in this form; each set of hierarchical pages is called a "model", and more than one model may reference a single database.

As noted previously, Design/IDEF provides for drawing IDEF-1X data model diagrams, but does not extract diagram information for any database processing by a user.

Distinguishing Capabilities

The merging of IDEF-0 diagramming with HCPNs is unique among CASE tools so far as the reviewers can determine. Petri nets are designed to represent highly concurrent processes with limited resources. HCPNs are an extension to Petri nets which add two well-known concepts, hierarchical decomposition and strong typing, and thus HCPNs should be a reasonable approach for representing complex systems. Petri nets and extensions such as HCPNs have a large body of formal mathematics to support them; [Rosenberg 87] contains 2074 entries. HCPNs appear to be a very reasonable approach for describing systems in which ambiguity cannot be tolerated, precise process synchrony is important, and/or where resource use is an important system factor [Davis 88]. The future analytical capabilities planned for Design/CPN are quite unusual among current CASE tools. As a potential minus, systems described using Petri nets tend to be difficult for applications-oriented non-computer experts to understand [Chiola 88, Jensen 81].

APPENDIX K

001

001 [Hamilton 86, Hamilton 88] refers to a toolset for modeling a system as a dynamic entity and also describing the same system as a static entity. The tool was developed by Hamilton Technologies, Inc., Cambridge, MA. AXES is an executable specification language available with 001, which combines functional and object-oriented design paradigms. 001 is commercially available on DEC VAX computers running Vax VMS 4.4 or higher, including the VAXstation. Hamilton Technologies is currently working to transport 001 to Sun workstations.

Graphics and Editing

001 includes graphics and text editors for building systems. In order to create diagrams, such as an object Type Map (TMap) for building a data model, the user selects options from the pull down menus, moves around a diagram via the cursor keys, and creates and labels design objects via keyboard input. The current version of 001 does not support the use of a mouse for editing. 001 automatically draws diagrams from their textual definition and generates a textual definition from a graphical diagram (i.e., the design is stored in the database as text).

User interface capabilities (e.g., fast browsing through system libraries) are provided along with the language generation tools called RATs (Resource Allocation Tools). A generic RAT is supplied with the 001. The user can build custom RATs using the generic RAT as a model template. The user then fills in the template as necessary. Documentation, reports, and source code are generated by this generalized RAT.

Design Semantics and Support

Systems in 001/AXES are defined in terms of functions, types, and structures. A function transforms its input typed objects to output typed objects using its children functions. A structure is used to specify how the children functions perform to support their parent in transforming its input to its output. The type of an object is defined in terms of a structure of children types. The decomposition of a type (called a parameterized type) defines the primitive functional interactions possible between a parent object and its children objects. These primitive functions are used in constructing more abstract functions. The functional aspect of a system is defined using a Function Map (FMap) which is a hierarchy of functions. The type (information or object modeling) aspect of a system is defined using a Type Map (TMap) which is a hierarchy of object types. Leaf node

functions form a data flow-like network where primitive functions transform objects from state to state. A TMap shows the network of relationships between primitive objects that may hold for a particular state.

The AXES model for a process is a function. In a 001/AXES model, all functions are formally related to each other in terms of their properties. One can explicitly designate a function or, once a function is scheduled in a particular implementation, it becomes a process. A 001/AXES model provides the necessary information for automatically allocating resources to specific functions as processes with varying implementation configurations. These configurations are dependent on the implementation environment's operating system, application language, and the designer's selected process structures.

Functions are decomposed into lower level functions using three primitive control structures or abstract control structures defined in terms of these primitive control structures which define the granularity of processes: OR, JOIN, and INCLUDE. The OR structure is used for decision control; that is, using the input of the parent (or decision) function, a partition function decides which child function is to be selected to perform for the parent. The INCLUDE structure is used to partition the parent's input into two disjoint sets of data objects which are then distributed to the children functions. The JOIN structure is used to express a dependence relationship between a parent and its two children. These structures are used in a Function Control Map (FMap), which defines the parent's behavior in terms of its children. The FMap is a network of relationships between instances of functions. The Type Map (TMap) is a network of relationships between objects.

001/AXES models a particular target system, its interoperative support systems, and their containing environment as functions. AXES has a non-procedural computational semantics that can be translated automatically into a procedural higher-order language. A Resource Allocation Tool (RAT) does this by mapping a 001/AXES functional and type specification onto some computational resource. Automatic code generating RATs are available in version 1 for C. FORTRAN and Ada TMap code generation capabilities are to be supported in the future.

Team Design Support

No special team design support is evident.

Documentation and Output

001 provides for automatic generation of code from system specifications, and allows the user to annotate the code via the RAT query and generation languages. The RoadMap of models in a library, TMaps, and FMaps may be printed using Postscript.

Documents may be generated by tailoring a generalized RAT.

Static Diagnostics

The principal analysis tool of the 001 environment is the Analyzer. It is used to ensure the consistency of typing and function interfaces. Time in a 001/AXES specification can be defined in more than one way. For example, a TIME data type can be actively used within the system itself for controlling timing or WHERE statements can be used to specify timing constraints about a 001/AXES language object such as a function. The amount of time a function (e.g., a single instance of an operation definition) takes to execute can be explicitly defined.

A system can be specified in 001/AXES to be a computer or resource architecture independent model by separating the functional (i.e., what is to be done) from the resource architecture. The functionality of a system is defined by the use of FMaps and TMaps. A designer specifies a resource architecture as a complex object also using FMaps and TMaps. The FMaps are used to define the construction procedure from which a resource architecture object will result.

A software/hardware system resource allocation mapping is made by specifying which elements of the functional architecture are to be executed on which elements of the resource architecture object.

Simulation

Simulations may be performed by several methods with 001. Currently, a system can be defined and automatically developed with the functional and resource architectures interwoven. In this case the primitive functions at the leaves of the FMaps are used to gather performance information during the execution of a system. The statements in 001/AXES relating to the static state of a system allow for a simulator to automatically monitor a system's dynamic behavior with respect to constraints. Although the dynamic definitions and some of the static descriptions of a 001 system are utilized in the automation of a 001 system, some of the static descriptions currently are used in a semi-automated mode. They serve as input to an automated fine-grained simulator.

A fine-grained event simulator is being developed to take advantage of the real-time semantics of 001/AXES, the description of the static entities of a system, and the separation between a functional system and its resource architecture system. A classical event model simulator prototype has been constructed. This simulator has been developed to automatically monitor some aspects of the dynamic behavior of certain kinds of 001 systems.

Adaptability

Hamilton Technology does not license the source code for 001/ AXES, nor does it permit a user to modify the generic RAT or the Analyzer. Hamilton Technology provides interface specifications so that a user can add to the CASE tool (e.g., add menu options). Hamilton Technology provides custom hooks (e.g., add a different graphics front-end to 001/AXES) for a fee.

Interoperability

001 provides limited interoperability on the basis of input or output data files. For example, the code generated by 001 can be exported to other tools or host environments. In addition, 001 permits the user to import data for simulation support.

Traceability

Tracing is limited to simple decomposition of objects.

Information Modeling Support

001 supports inheritance. For example, in a TMap, a hierarchical system of data types can be defined. Types at the nodes of a TMap inherit their behavior in terms of primitive operations from the parameterized type (or structure) used for their decomposition. Each type inherits the properties of its subtypes.

Distinguishing Capabilities

001/AXES captures system behavior and resource allocations. Simulation from functional specifications is also a distinguishing capability.

APPENDIX L

FORESIGHT

Foresight is a front-end CASE tool for specifying and analyzing system requirements via simulation and animation. Foresight is a product of Athena Systems, Inc., of Sunnyvale, California. It currently runs on the Sun/3, Sun/4, the Sun SPARC workstations, and the HP 9000 series. The system is written in C++, and the user interface resides on top of X Windows. Athena Systems is currently porting Foresight to Apollo and DEC workstations [Athena Systems 89].

Graphics and Editing

Foresight provides both graphic and text editors. The user-interface to these editors resembles the standard user interface of the Apple Macintosh. The types of graphic editors supported by Foresight differ from those found in most of the data flow oriented CASE tools in that the tool provides a general-purpose graphics editor similar to StP's PICture editor.

The data flow diagram and state transition diagram editors provide a very extensive library of modeling constructs. In addition to annotation elements, the data flow diagram editor provides has an extended palette of design objects, allowing the user to differentiate between different types of data flows (discrete vs. continuous), prompts (e.g., trigger, enable, disable, suspend, resume), and so on. The state transition editor provides for graphical annotations, as well as initial state, state, transition, I/O block, and reshaper design objects on the menu.

Design Semantics and Support

Foresight is similar to i-Logix's Statemate, in that both tools can analyze and predict the behavior of real-time systems from textual and graphical specification of system requirements, and simulation and animation of the requirements. Foresight and Statemate provide both functional and constraint modeling tools. Both tools' notations are based on Structured Design/Structured Analysis with real-time extensions. The major difference between Foresight and Statemate is notation. Both use a form of data flow diagrams, but Foresight relies on state transition diagrams, whereas Statemate uses state charts, a more powerful notation. Neither tool has come up with a solution to the problem of representing and simulating replicated processes. Foresight does have a concept called a reusable process. A predefined set of these processes are made available to the user, and user-defined processes may be added to the library. The reusable process

notation does not necessary solve the problem of notating replication since for each instance of a replicated process that is required, the user must copy and paste the corresponding reusable library element.

Foresight's constraint modeling capability can specify timing constraints between a system's components and external events. In addition, Foresight's large set of modeling constructs makes it possible to specify and simulate software, hardware, and firmware systems with the same tool.

Foresight's formal notation can be used to model both primitive and composite data types. Low-level data elements are modeled with primitive data types. Foresight's primitive data types are: boolean, real, integer, complex, enumerated, and string. Composite data types are modeled as arrays and records.

The tool provides support for the description and specification of real-time systems via extensions to SA/SD. In addition, Foresight supports the ESML standardization effort (i.e., standards for real-time system specification). The tool also provides extensions to ESML for use in simulating a real-time system.

Three types of data flows are available for use in modeling a system. Continuous flows are flows that are available over an interval of time. Discrete flows are only available at a point in time. Control flows represent messages that are used to activate or deactivate a process. All three types of flows are used by Foresight's simulation facility.

Processes can be decomposed into state transition diagrams and mini-specs. These three components can be used to define abstract processes.

The tool provides a predefined set of reusable functions and operations, called Library Elements. These functions and operations are executable, and fall into five major categories:

a. Math and logic (e.g., absolute value, sin, cos, adder)

b. Signal processing (e.g., hold, sample, Z delay)

c. Timing and validation (e.g., event, max delay, elapsed time)

d. Data manipulation (e.g., splitter, merger, queue)

e. Electronic I/O panel (i.e., I/O devices)

The user may create additional functions and operations as needed. User-defined library units will not be used by the simulation tool unless the user modifies Foresight's source code.

Team Design Support

Foresight relies on the host machine to provide configuration and file management facilities.

Documentation and Output

The graphic and text editors support PostScript formatted output. Templates are available for outputting specification and design data in MIL-STD-2167A documentation format.

Static Diagnostics

Foresight provides a static analyzer for checking the syntax of processes and the decomposition of processes, including low-level objects such as mini-specs.

Simulation

Foresight provides the user with an interactive, discrete-event simulator. Values for a system's data flows can be entered by the designer during a simulation. The code that is generated to run the simulation can be viewed by the user, and exported to other development tools for implementation of the system. Output during the simulation can be viewed in either graphical or textual form. The electronic I/O panel elements are used to drive the simulation of a system.

The simulator supports animation, stepping through a simulation, as well as three running modes: step, fire, and advance. The simulation can be stopped and restarted at any time, allowing for further development of a model before continuing the simulation of a system.

Mini-specs are executable by the simulator. The specifications use a formal syntax which is a subset of the Ada programming language, including: assignment, conditional branching, case, while, and for looping constructs. Athena Systems is currently working on a bi-directional translator to and from Ada, in order to further facilitate the reverse engineering process (i.e., import existing code for analysis).

Interoperability with Other Tools

Foresight stores specification and design information in a proprietary object management system (OMS). The data formats used by Foresight's OMS are published, permitting the user to import and export data. The granularity at which objects are stored can be specified by the user.

Traceability

Foresight does not support traceability of design data.

Information Modeling Support

Foresight supports inheritance of properties by design objects (e.g., processes). Foresight does not have an ERA diagram editor, nor does it have a view mechanism for its underlying database. Foresight provides a formal set of rules for the propagation of data flows throughout the hierarchy of data flow diagrams. Furthermore, rules are provided for the execution of processes. For example, a logical "AND" process must have all its inputs before firing, and a logical "OR" process will fire when any inputs are available. If control flows come into a process, the process must be activated before it can be executed. Upon firing, the underlying description and the process are executed by Foresight's simulation/animation tools.

Distinguishing Capabilities

The distinguishing characteristics of Foresight are that it supports interactive simulation and animation, specification of reusable components, and constraint and timing modeling.

APPENDIX M

VIRTUAL SOFTWARE FACTORY

The Virtual Software Factory (VSF) is a meta-CASE tool. It is a product of Systematica Limited, Bournemouth, England. VSF currently runs on the Sun 3 and 4 series workstations and the DEC VAXstation 3000 and 3100 series. A version is planned for the IBM PS/2 series. The philosophy underlying VSF is that integration should occur at the information level rather than at the tool level. This does not imply that VSF could not be integrated into another framework. VSF addresses two types of integration: method and design database integration. The assumption underlying this tool is that only a minimal standard for integration is needed. The goal is to minimize the overhead associated with adding and deleting tools from a Software Engineering Environment (SEE). VSF is intended to be flexible enough that methodologies supported by a tool can be added or deleted from the SEE. Tools can be rapidly developed to work with existing methodologies or to implement a new methodology. VSF is a meta-CASE tool in that the user can define a CASE tool based on the methodology required to be used for a software project. VSF is a workbench, not a SEE, the difference being that a workbench is a set of tools that are not integrated, whereas the tools in a SEE are all integrated. VSF provides for verifiability, tailorability, and traceability across the entire life cycle. VSF consists of two tool sets: Methods Engineering Workbench (VSF/MWB), and Analyst Workbench (VSF/AWB).

Graphics and Editing

VSF provides for graphical and textual editing. The methods workbench is primarily textually-oriented. The MWB is used to define the graphics environment for the workbench to be used by the software engineers. The SWB supports the use of a mouse, menus, and graphics. How these capabilities can be accessed is determined by the methods engineer when he writes the rules implementing a particular methodology.

Design Semantics and Support

Methods Engineering Workbench

The MWB is used to define methodologies and configure the design environment in which a software engineer, referred to in the Systematica literature as an analyst, must work within for a particular project or set of projects. The assumption underlying this tool is that an organization has a full-time methodologist, referred to as a methods engineer,

that is responsible for defining and maintaining in VSF the various methodologies supported by the organization. The MWB provides two types of configurability. The methods engineer first uses the MWB to define the conceptual model for a methodology, consisting of the structure (i.e., types) and rules (i.e., interaction between types). The rules provide a "filter" mechanism for performing syntax and semantic checks.

The second type of configurability supplied is the definition of notations for a methodology. The methods engineer configures the design editors via the notation and syntax definitions. This is done by defining a set of templates that control the presentation of text and graphics, including documentation standards for the methodology.

Analyst Workbench

The AWB consists of the set of graphical and textual editors that were predefined for methodologies in the MWB. These editors provide for entering, storing, updating, and viewing design information. A database browser, similar to that of TRW's DCDS, is provided for use in making ad hoc queries of the design and methods databases. Documents are stored and retrieved using a hypertext approach. A document is a piece of the design defined by the user (i.e., the level of granularity is determined by the methods engineer). VSF has a built-in file manager.

Changes to the methodology made via the MWB are propagated throughout the design databases associated with that methodology. Syntax and semantics checks are done on-line, and additional checks may are invoked by the analyst.

VSF can automatically generate code from a design if code generation templates have been predefined by the methods engineer. The completeness of the code depends on the methodology used. Systematica's implementation of the HOOD methodology in VSF was demonstrated. We were able to graphically design a multi-sensor multi-tasking system using the HOOD methodology, and then generate the corresponding Ada source code. In addition, we were able to modify the HOOD methodology rule base (i.e., the filter mechanism) using the MWB. We then verified that the VSF filter that performs rule checking actually propagated and enforced the rules we specified.

Two methodologies that have been implemented using VSF and are available as off-the-shelf tools are SSADM and HOOD/Ada. SSADM is a British government standard for EDP system development. HOOD is a hierarchical object-oriented design method and has become a de facto standard for European aerospace Ada development. Other methods have been implemented by Systematica and VSF users, including CORE, a requirements capture method used in some European aerospace organizations, and Mascot 3/Ada, a British Defense Ministry standard for real-time systems development.

The methods engineer defines the granularity which is required to conduct configuration management and project management. That is, the user defines "fragments" of a design, independent of units of documentation, which logically should be managed as one piece. The user uses the configuration and project management tools available in the host environment. The host environment is a shell around VSF.

Formalism

The formalism underlying VSF is a decidable second-order logic. This logic allows the user to specify:

- Predicate existence through naming
- Documents as a set of attribute grammars
- Describe context sensitive grammars (i.e., types and properties)
- Explicit quantification

The methods engineer is responsible for specifying concepts, and must already be familiar with the logic formalisms required to do so. VSF comes with a high-level, internal logic specification language that resembles PROLOG (the language does perform PROLOG-like operations such as pattern matching). VSF itself, however, is written in Ada, and consists of approximately 300,000 lines of source code.

VSF also supports beliefs, belief generation rules, pre- and post- conditions, etc. Each of these items can be associated with a particular state.

Team Design Support

The AWB does not provide for multi-user design. Instead, it does provide for merging of design information into a central design database via the VSF merge facility.

Documentation and Output

The methods engineer specifies the documentation, such as MIL-STD-2167A, required by a methodology via the MWB. The design databases created via VSF are stored in a VSF-specific format. VSF can also produce an ASCII version of a database, as well as a printable form. For graphics, VSF supports Postscript, Interleaf, and HPGL standard data formats.

Static Diagnostics

The methods engineer, using the filter mechanism, implements the checking rules for static diagnostics used by the analyst.

Simulation

No capability for design simulation or dynamic checking is provided by VSF.

Adaptability

The underlying concept of VSF is as an adaptable environment, as described in the introduction.

Interoperability with Other Tools

VSF has a data import/export facility. Any design fragment can be "conserved" to another tool, whose output can then be "merged" (with con ıict checking) back into the VSF workbench.

Traceability

The methods engineer can define a traceability model between design objects of earlier or later project phases. VSF can treat individual clauses of a document, such as a contract document or RFQ, as a design object. Traceability links can be created between these objects. Successive links could be created to go through to code generation.

Information Modeling Support

VSF support meta-modeling constructs such as multiple inheritance across hierarchies, multiple design databases, automatic translation between methodologies, and specification and enforcement of rules for methodologies. The amount of semantics incorporated into the modeling are only limited by the limitations of the underlying VSF formalisms (logic constructs). Schema(s) can be described using the VSF formalisms.

Distinguishing Capabilities

VSF represents a different approach from most CASE tools. It can be considered as a meta-CASE tool for building methodologies and for interactively prototyping methodologies.

APPENDIX N

ADAGEN

Adagen is a tool produced by Mark V Systems Limited of Encino, California. Adagen is primarily a single general-purpose graphical editor which can be tailored for specific notations, and incl ıdes support for a number of notations. Adagen also includes a program to generate Ada code from Buhr diagrams and a reverse engineering program to create Buhr and compilation dependency diagrams from Ada code. Adagen operates on many platforms, including IBM-PC compatibles (under Microsoft Windows), Apple Macintosh, Sun 3 and SPARCstation, Hewlett-Packard 9000/300 series, Apollo 2000, 3000, and 4000 families, Data General AViiON, MIPS, and the DEC Vaxstation under VMS. The IBM-PC DOS version uses Microsoft Windows, the Macintosh uses the Multi-Finder system, and all other platforms use X-Windows.

Graphics and Editing

The system had good performance on the IBM-PC but was relatively slow on a Sun under the X-Windows system, possibly due more to X-Windows itself than Adagen. The graphical editor is intuitive and easy to use and includes a one-level undo. Online help displays are available.

Design Semantics and Support

Adagen supports the graphical editing of many different design notations, in particular a number of object-oriented notations. These include Booch diagrams (1985 version), Buhr diagrams (1984 and 1990 versions), Object Oriented Software Development (OOSD) by Ed Colbert (not be confused with OOSD by Anthony Wasserman), Object Oriented Requirements Analysis (OORA) by Peter Coad, Ada box structures by Ed Comer, and Don Firesmith's Ada Development Method notation. Adagen also supports editing "traditional" notations such as data flow diagrams (Yourdon/DeMarco and Gane/Sarson) with Ward/Mellor real-time extensions, state transition diagrams, data model diagrams, and structure Constantine charts. Adagen also support Rnets and Fnets (from DCDS), functional flow diagrams, James Martin's action diagrams, and Chen's Entity Relationship Attribute notation.

Team Design Support

Adagen provides no team design support.

Documentation and Output

Depending on the platform, Adagen can produce output of its text and graphics in Ventura Publisher, Aldus PageMaker, Unix pic, Apple .pict metafile, Interleaf, Framemaker, or Postscript format. There are no forms for standard documents such as MIL-STD-2167A included with the tool.

Static Diagnostics

Each diagram type may have a rule specifying the consistency required between a diagram and its parent. This rule is checked when a diagram is loaded into the editor. Currently there are two possible choices for this rule, "supra-constrained" or "interface-constrained." In a diagram type which is "supra-constrained" the input and output flows of a child diagram must be the same as its parent object. Data flow diagrams, for example, are "supra-constrained." In a diagram type which is "interface-constrained" the outermost graphical object must have the same interfaces as the higher level object and the arcs are not conserved. Buhr diagrams, for example, are "interface-constrained."

Version 1.6 includes some checks, particularly for Buhr diagrams, for checking a diagram's contents interactively. These cannot be modified by the user in version 1.6. Mark V states that version 1.7 will use a special language to perform these interactive checks and thus allow users to create their own checks. There is no batch processing mode for checking a set of diagrams, either for the diagram contents or for consistency with the diagram parent.

Simulation

Adagen has no simulation capability per se. Adagen can generate Ada code from Buhr diagrams via a Prolog program. Source for this program is available under separate license from Mark V.

Adaptability

Menu options and accelerator keys can be modified by users, and each menu option or key can choose a new icon or cause certain tool operations (termed "atoms") to

be executed. The transformations to and from Ada can be tailored by a knowledgeable user. Mark V states that version 1.7 of the tool will have semantic binding of icons to database entities through BNF-like rules, allowing users to define their own constraint checking rules, and arbitrary programs to be executed instead of the set list of atoms. Mark V appears to be working to make their tool more extensible by knowledgeable users. Users cannot create their own atoms now, but the vendor states that version 1.7 will have this capability.

Interoperability

On the IBM-PC Adagen may be used both as a client and as a server using the Microsoft Dynamic Data Exchange (DDE) capability. Thus Adagen on the IBM-PC may consider other tools as leaves of its database (using other programs such as SQL servers to access the data) and is fully operable by other programs. This capability is expected to be available on the Apple Macintosh when the System 7 message capability is available. Mark V states that it would be easy to add similar capabilities to Adagen using TCP/IP if customers desire.

Traceability

Any Adagen object may be linked to any other object. There is no typing of the link, though some typing can be inferred by the type of the diagram containing object. This linking is done through an extremely simple mechanism: an object is selected (on one diagram) and Adagen puts its object identifier on a list. Other objects on other diagrams may be selected and linked to the first object.

Information Modeling Support

Version 1.6 only edits diagrams and does not support a separate database. Mark V states that version 1.7 will include an ERA database with some object-oriented extensions (such as inheritance) and a specialized Prolog-like language for entering rules.

Distinguishing Capabilities

Adagen's distinguishing capabilities are its extensibility, Ada code generation and reverse-engineering features.

Adagen can reverse-engineer existing Ada code into compilation dependency diagrams and Buhr diagrams. The appearance of the resulting diagrams can be tailored

by a user, and in fact such tailoring is necessary to produce legible diagrams. Iterative cycling between text and graphics is not supported, however, because some Ada constructs are not entered back into the diagrams and a large number of settings must be individually adjusted. This capability does, however, appear promising.

APPENDIX O

ACRONYMS

Term	Meaning	Associated Tool
ADAS	Architecture Design and Assessment System	Teamwork
ASA	Advanced System Architectures	Auto-G
AWB	Analyst Workbench	VSF
CASE	Computer-aided Software or Systems Engineering tool	
CCC	Change Configuration Control	
CDIF	CASE Design Interchange Format	
DBMS	Database Management System	
DCDS	Distributed Computing Design System	DCDS
DDL	Distributed Design Language	DCDS
DDM	Distributed Design Method	DCDS
DEC	Digital Equipment Corporation	
DOD	Department of Defense	
DPI	Document Production Interface	Teamwork
DPS	Document Preparation System	StP
E-R	Entity-Relationship	
ERA	Entity, Relationship, Attribute Model	
FMap	Function Map	001
Fnet	Function Network	DCDS, RDD
HCPN	Hierarchical Colored Petri Net	Design/family
IBM	International Business Machines	
ICAM	Integrated Computer-Aided Manufacturing	
IDE	Interactive Development Environments	StP
IDEF	ICAM Definition (method)	
ISAM	Indexed Sequential Access Method	
Inet	Item Network	DCDS, RDD
MCM	Model Configuration Management	Teamwork
MDL	Module Development Language	DCDS

MDM	Module Development Method	DCDS
MWB	Methods Engineering Workbench	VSF
OAE	Object Annotation Editor	StP
OMS	Object Management System	
PC	Personal Computer	
RATs	Resource Allocation Tools	001
RDD	Requirements Driven Design	RDD
RSL	Requirements Specification Language	DCDS
RVTS	Requirements Verification Tool Set	TAGS
SADMT	SDI Architecture Dataflow Modeling Technique	
SC	Simulation Compiler	TAGS
SDI	Strategic Defense Initiative	
SDIO	Strategic Defense Initiative Organization	
SDL	SDI System Description Language	
SEE	Software Engineering Environment	
SQL	Structured Query Language	
SREM	Software Requirements Engineering Method	DCDS
SSL	System Specification Language	DCDS, RDD
SYSREM	System Requirements Engineering Method	DCDS
StP	Software through Pictures	StP
TAGS	Technology for the Automated Generation of Systems	TAGS
TMap	Type Map	001
TSL	Test Support Language	DCDS
TSM	Test Support Method	DCDS
USASDC	U.S. Army Strategic Defense Command	DCDS
VHDL	VHSIC Hardware Description Language	
VHSIC	Very High Speed Integrated Circuit	
VSF	Virtual Software Factory	VSF

References

[Athena Systems 89] Athena Systems, Inc. 1989. Foresight: Modeling and Simulation Toolset for Real-Time System Development, Athena Systems, Inc, Sunnyvale, CA.

[Chiola 88] Chiola, G., Bruno, G., and Demaria, T. 1988. Introducting a color formalism into generalized stochastic Petri nets. Proceedings of the 9th European Workshop on Applications and Theory of Petri Nets, Venice, pp. 202-215.

[Cohen 87] Cohen, Howard, et al., 1987. SDI Architecture Dataflow Modeling Technique (SADMT) Simulation Framework. IDA Paper P-2036 (Draft), Institute for Defense Analyses, Alexandria, VA.

[Coolahan 83] Coolahan, Jr, James E, and Roussopoulos, Nicholas. 1983. Timing Requirements for Time-Driven Systems Using Augmented Petri Nets. IEEE Transactions of Software Engineering, Vol. SE-9, No. 5, pp. 6003-6616.

[Davis 88] Davis, Alan M. 1988. A Comparison of Techniques for the Specification of External System Behavior. Communications of the ACM, pp. 1098-1115.

[Fife 87] Fife, D. (Ed.). 1987. Evaluation of Computer-Aided System Design Tools for SDI Battle Management/C3 Architecture Development. IDA Paper P-2062, Institute for Defense Analyses, Alexandria, VA.

[Hamilton 86] Hamilton, M. H. 1986. Zero-Defect Software: The Elusive Goal. IEEE Spectrum, pp. 48-53.

[Hamilton 88] Hamilton, M. H., and Hackler, R. 1988. The Realization of Ultra Reliable Models, Simulations and Software Systems. Hamilton Technologies, Inc.

[Harel 87] Harel, David. 1987. Statecharts: a Visual Formalism for Complex Systems, Science of Computer Programming, pp. 231-274.

[Harel 88] Harel, David. 1988. STATEMATE: A Working Environment for the Development of Complex Reactive Systems. Pro. 10th IEEE International Conference on Software Engineering.

[Harel 90] Harel, David, Lachover, H., Naamad, A., Pnueli, A., Politi, M., Sherman, R., Shtull-Trauring, A., and Trakhtenbrot, M. 1990. STATEMENT: A Working Environment for the Development of Complex Reactive Systems. IEEE Transactions on Software Engineering, Vol. 16, No. 4.

[Harper 86] Harper, Robert, MacQueen, David, and Milner, Robin. 1986. Standard ML Technical Report ESC-LFCS-86-2, University of Edinburgh, LFCS, Department of Computer Science, University of Edinburgh, The King's Buildings, Edinburgh EH9 3JZ.

[IDE 88] Interactive Development Evnironments. Software through Pictures Release 4.0 User Manual (Sun Version), San Francisco, CA.

[Jensen 81] Jensen, Kurt. 1981. Coloured Petri Nets and the invariant-method. Theoretical Computer Science, No. 14, pp. 317-336.

[Linn 88] Linn, J. L., et al. 1988. Strategic Defense Initiative Architecture Dataflow Modeling Technique Version 1.5. IDA Paper P-2035, Institute for Defense Analyses, Alexandria, Va.

[Marsan] Marsan, M. A. and Chiola, G. On petri nets with deterministic and exponential firing times, in [Rosenberg]. pp. 104-132.

[Rosenberg 87] Rosenberg, G. (Ed.). 1987 Advances in Petri Nets. Lecture Notes of Computer Science, Vol. 266, Springer-Velag.

Distribution List for IDA Paper P-2177

NAME AND ADDRESS	NUMBER OF COPIES
Sponsor	
Lt. Col. James Sweeder Program Manager, Software Engineering SDIO/ENA The Pentagon Room 1E149 Washington, DC 20301-7100	3
Other	
Defense Technical Information Center Cameron Station Alexandria, VA 22314	2
Mr. Karl H. Shingler Department of the Air Force Software Engineering Institute Joint Program Office (ESD) Carnegie Mellon University Pittsburgh, PA 15213-3890	1
Chuck Lillie SAIC 1710 Goodridge Drive P.O. Box 1303 McLean, VA 22102	1
Jack Kleinert SAIC 1710 Goodridge Drive P.O. Box 1303 McLean, VA 22102	1
Joyce Lytle SAIC 1710 Goodridge Drive P.O. Box 1303 McLean, VA 22102	1

NAME AND ADDRESS	NUMBER OF COPIES
Mike Mitrione Dynamics Research Corp. 1755 Jefferson Davis Hwy Suite 802 Arlington, VA 22202	1
Hobart Mendenhall CAE Link 11800 Tech Road Silver Spring, MD 20904	1
Ken Finscher Teledyne Brown Engineering Stop 56 Cummings Research Park P.O. Box 070007 Huntsville AL 35807-7007	1
Debra Yeagle Naval Surface Warfare Ctr. Code F31 Dahlgren, VA 22448-5000	1
Hui Huang NIST	1
Bldg 220, Room B124 Gaithersburg, MD 20899	1
John Salasin GTE Govt. Systems Corp. 1700 Research Blvd Rockville, MD 20850	1
LCDR S.M. Vause COMNAVSUPSYSCOM Code 0482 Washington, DC 20376	1
Les Williams Ascent Logic Corporation 180 Rose Orchard Way, Suite 200 San Jose, CA 95134	1
Jim Long Omnitech Systems 2070 Chain Bridge Road, Suite 320 Vienna, VA 22182	1

NAME AND ADDRESS	NUMBER OF COPIES
Virgina P. Kobler Chief, Technology Branch Battle Management Division US Army Strategic Defense Command P.O. Box 1500 Huntsville, AL 35807	1
Mike Guillebeau TRW 213 Winn Drive Huntsville, AL 35805	1
Miguel Carrio 3700 Pender Drive Suite 200 Fairfax, VA 22030	1
Joe Fox Software Architecture and Engineering, Inc. 1600 Wilson Blvd. Suite 500 Arlington, VA 22209	1
Christopher Williams Advanced System Architectures, LTD 4550 Forbes Blvd. Lanham, MD 20706	1
Dr. Anthony Wasserman, President Interactive Development Environments, Inc. 150 Fourth Street, Suite 210 San Francisco, CA 94103	1
Ms. Yvonne Cekel Marketing Manager Cadre Technologies, Inc. 222 Richmond Street Providence, RI 02903	1
Joseph Weeks I-Logix Inc. 1101 King Street, Suite 601 Alexandria, VA 22314	1
Lawrence Markosian Reasoning Systems, Inc. 1801 Page Mill Rd, Suite 125 Palo Alto, CA 94304	1

NAME AND ADDRESS	NUMBER OF COPIES
Robert Seltzer Meta Software Corporation 150 Cambridge Park Drive Cambridge, MA 02140	1
Dr. Margaret Hamilton Hamilton Technologies, Inc. 17 Inman Street Cambridge, Massachusetts 02139	1
Ted Liu Athena Systems, Inc. 139 Kifer Court Sunnyvale, CA 94086	1
Richard Iliff SDIO/ENS The Pentagon Washington, DC 20301-7100	1
Michael D. Henry Flight Command and Data Mgmt Systems Section Jet Propulsion Laboratory California Institute of Technology 4800 Oak Grove Drive Pasadena, CA 91109-8099	1
Gary D. Sheets Lockheed Space Station Freedom - SSE System Project 1150 Gemini Avenue Houston, TX 77058	1
Axel H. Ahlberg GE Aerospace National Test Facility, MS-N9020 Falcon AFB, CO 80912-5000	1
Shmuel Halevi Technology Research Group 2 Park Plaza Suite 570 Boston, MA 02116	1
Robert McCauley Martin Marietta Falcon AFB, CO 80912-5000	1

NAME AND ADDRESS	NUMBER OF COPIES
Lt. Col. John Morrison NTBJPO Falcon AFB, CO 80912-5000	1
Chuck Howell M/S Z645 The MITRE Corporation 7525 Colshire Drive McLean, VA 22102-3481	1
Joel Van Berkum Electronic/Software Engineer OO-ALC/MMETI Hill AFB, UT 84056-5609	1
CSED Panel Dr. Dan Alpert, Director Program in Science, Technology&Society University of Illinois Room 201 912-1/2 West Illinois Street Urbana, Illinois 61801	1
Dr. Thomas C. Brandt 10320 Bluet Terrace Upper Marlboro, MD 20772	1
Dr. Ruth Davis The Pymatuning Group, Inc. 2000 N. 15th Street, Suite 707 Arlington, VA 22201	1
Dr. C.E. Hutchinson, Dean Thayer School of Engineering Dartmouth College Hanover, NH 03755	1
Mr. A.J. Jordano Manager, Systems & Software Engineering Headquarters IBM Federal Systems Division 6600 Rockledge Dr. Bethesda, MD 20817	1

NAME AND ADDRESS	NUMBER OF COPIES
Dr. Ernest W. Kent Philips Laboratories 345 Scarborogh Road Briarcliff Manor, NY 10510	1
Dr. John M. Palms, President Georgia State University University Plaza Atlanta, GA 30303	1
Mr. Keith Uncapher University of Southern California Olin Hall 330A University Park Los Angeles, CA 90089-1454	1
IDA	
General W.Y. Smith, HQ	1
Mr. Philip L. Major, HQ	1
Dr. Robert E. Roberts, HQ	1
Ms. Ruth L. Greenstein, HQ	1
Ms. Anne Douville, CSED	1
Dr. Richard J. Ivanetich, CSED	1
Mr. Terry Mayfield, CSED	1
Ms. Sylvia Reynolds, CSED	2
Dr. Richard Wexelblat, CSED	1
Mr. David Wheeler, CSED	15
Dr. Dennis Fife, CSED	50
Mr. Edgar Sibley, CSED	1
Mr. J. Bret Michael, CSED	1
IDA Control & Distribution Vault	3